Integer Codes Correcting High-Density Byte Asymmetric Errors

Aleksandar Radonjic and Vladimir Vujicic

***Abstract*—In optical networks without optical amplifiers the number of received photons never exceeds the number of sent ones. Hence, upon transmission, only asymmetric (1 → 0) errors can occur. Motivated by this fact, in this letter we present a class of integer codes capable of correcting high-density asymmetric errors within a *b*-bit byte. Unlike classical codes, these codes use integer and lookup table operations. As a result, they can be implemented "for free", i.e. without modifying the network hardware.**

***Index Terms*—Integer codes, error correction, asymmetric errors, look-up table.**

I. Introduction

Error control codes are usually designed for use on binary symmetric channels, where the error probabilities 0→ 1 and 1→ 0 are equal. However, in certain systems, the errors have a highly asymmetric nature. For the purpose of this paper, the most interesting example are optical networks without optical amplifiers (ONWOAs) (e.g. local and access networks) [1]. In these networks, the number of received photons never exceeds the number of transmitted ones. Hence, upon transmission only asymmetric (1 → 0) errors can occur [2], [3]. Besides this, it is known that these errors affect small number of bits. More precisely, the experiments showed that 99% to 99.9% of all errors are *t*-bit errors ($1 \le t \le 4$) confined to one or two adjacent bytes [4]-[6].

Another common feature of ONWOAs is high computing power of network nodes. This can be seen from the fact that exterior nodes (e.g. PCs and servers) contain general purpose processors, whereas interior nodes (e.g. switches and routers) are always equipped with network processors (NPs) [7], [8]. The only difference between these chips is that NPs integrate coprocessors for common kernels of computation (e.g. lookup and cryptographic operations). On the other hand, both these chips have integer execution units as well as the memory system including the caches [7], [8]. Hence, it can be said that they are designed for integer and lookup table operations.

Motivated by these facts, in this letter we present a new class of integer codes. The proposed codes, like those in [9]-[11], have several desirable properties including systematic structure, simple encoding/decoding procedures and fast error correction algorithm based on table lookups. However, unlike [9]-[11], the codes presented in this paper can correct two types of errors within a *b*-bit byte: single *t*/*d* asymmetric errors and double adjacent *t*'/*d* asymmetric errors, where $1 \le t' < t < d$ and $p = b/d \ge 2$. Thanks to this feature, they are more suitable for use in ONWOAs than the codes proposed in [9]-[11].

The organization of this paper is as follows: Section 2 deals with the construction of integer codes capable of correcting single *t*/*d* and double adjacent *t*'/*d* asymmetric errors in a *b*-bit byte (integer $(S_{t/d}AEC\text{-}DA_{t'/d}AEC)_b$ codes) Section 3 explains the implementation strategy for these codes, while Section 4 concludes the letter. Table 1 shows the notations used in this work.

TABLE I
NOTATIONS USED IN THIS LETTER.

Symbol	Meaning
B_i	Integer value of the *i*-th *b*-bit data byte at the sender side
C_B	Integer value of the *b*-bit check-byte at the sender side
$\hat{B}_i$	Integer value of the received *i*-th *b*-bit data byte
$\hat{C}_B$	Integer value of the received *b*-bit check-byte
$C_{\hat{B}}$	Integer value of the *b*-bit check-byte at the receiver side

II. Codes Construction

A. Encoding and Decoding Procedures

Let $Z_{2^b-1} = \{0, 1, \ldots, 2^b - 2\}$ be the ring of integers modulo $2^b - 1$, and let C_i and C_{k+1} be integers such that $C_i \in Z_{2^b-1} \setminus \{0,1\}$ and $C_{k+1} = -1$. Now, suppose that the data are divided into *k* *b*-bit bytes. In that case, the encoder will compute the check-byte in the same way as in [9]-[11], i.e. by using the following operations:

$$C_B = [C_1 \cdot B_1 + \cdots + C_k \cdot B_k] \pmod{2^b - 1} = \sum_{i=1}^{k} C_i \cdot B_i \pmod{2^b - 1} \quad (1)$$

At the receiver, the decoder will perform the same calculation

$$C_{\hat{B}} = [C_1 \cdot \hat{B}_1 + \cdots + C_k \cdot \hat{B}_k] \pmod{2^b - 1} = \sum_{i=1}^{k} C_i \cdot \hat{B}_i \pmod{2^b - 1} \quad (2)$$

after which the syndrome *S* will be formed

$$S = [C_{\hat{B}} - \hat{C}_B] \pmod{2^b - 1} \quad (3)$$

Obviously, when $S \neq 0$, the codeword is corrupted by one or more errors. Whether these errors can be corrected or not depends on the values of the coefficients C_i. On the other hand, the coefficient C_{k+1} always has the same value, since it corresponds to the errors within the check-byte.

The authors are with the Institute of Technical Sciences of the Serbian Academy of Sciences and Arts, 11000 Belgrade, Serbia (e-mail: sasa_radonjic@yahoo.com; vujicicv@yahoo.com).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2016.2644661, IEEE Communications Letters

B. Necessary and Sufficient Conditions

Definition 1. *An error is called t/d asymmetric error if t or fewer bits in a d-bit byte are in a 1 → 0 error, where $1 \le t < d$.*

Definition 2. *An error is called low-density byte asymmetric (LDBA) error if there exists one t/d asymmetric error within a b-bit byte, where $p = b/d \ge 2$.*

Definition 3. *An error is called high-density byte asymmetric (HDBA) error if there exists two adjacent t'/d asymmetric errors within a b-bit byte, where $1 \le t' < t < d$ and $p = b/d \ge 2$.*

To make these definitions more clear, we give examples of LDBA and HDBA errors (Fig. 1).

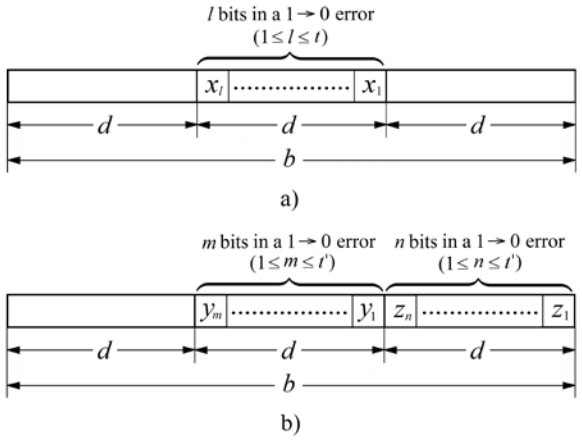

Fig. 1. Examples of (a) LDBA errors and (b) HDBA errors, where $p = 3$.

Definition 4. *Let $0 \le x_1 < \cdots < x_l < d$, $1 \le l \le t$, $0 \le r \le p-1$, and let $e_{l,r} = \{(2^{x_l} + \cdots + 2^{x_1}) \cdot 2^{d \cdot r}\}$ be the difference between the integer values of the correct b-bit byte and its received erroneous counterpart affected by LDBA error. Then, the set of syndromes corresponding to LDBA errors is defined as*

$$s_1 = \left\{ \bigcup_{i=1}^{k+1} \bigcup_{l=1}^{t} \bigcup_{r=0}^{p-1} \left(-C_i \cdot e_{l,r} \right) \left(\mathrm{mod}\ 2^b - 1 \right) \right\} \quad (4)$$

Definition 5. *Let $0 \le y_1 < \cdots < y_m < d$, $0 \le z_1 < \cdots < z_n < d$, $1 \le m,n \le t'$, $0 \le s \le p-2$ and let $e_{m,n,s} = \{(2^{y_m} + \cdots + 2^{y_1}) \cdot 2^{d \cdot (s+1)} + (2^{z_n} + \cdots + 2^{z_1}) \cdot 2^{d \cdot s}\}$ be the difference between the integer values of the correct b-bit byte and its received erroneous counterpart affected by HDBA error. Then, the set of syndromes corresponding to HDBA errors is defined as*

$$s_2 = \left\{ \bigcup_{i=1}^{k+1} \bigcup_{m=1}^{t'} \bigcup_{n=1}^{t'} \bigcup_{s=0}^{p-2} \left(-C_i \cdot e_{m,n,s} \right) \left(\mathrm{mod}\ 2^b - 1 \right) \right\} \quad (5)$$

Now, we can prove the following theorem.

Theorem 1. *The codes defined by (1)-(5) can correct all LDBA and HDBA errors iff there exist k mutually different coefficients $C_i \in Z_{2^b-1} \setminus \{0, 1\}$ such that*

1. $|s_1| = (k+1) \cdot p \cdot \sum_{l=1}^{t} \binom{d}{l}$
2. $|s_2| = (k+1) \cdot (p-1) \cdot \left[\sum_{n=1}^{t'} \binom{d}{n} \right]^2$
3. $s_1 \cap s_2 = \varnothing$

where $|A|$ denotes the cardinality of A, and $A \cap B$ the intersection of A and B.

Proof. Condition 1 of this theorem says that LDBA errors generate $(k+1) \cdot p \cdot \sum_{l=1}^{t} \binom{d}{l}$ syndromes that are nonzero. To prove this, observe that the set s_1 can be expressed as

$$s_1 = \bigcup_{i=1}^{k+1} M_i$$

where

$$M_1 = \left\{ \bigcup_{l=1}^{t} \bigcup_{r=0}^{p-1} \left(-C_1 \cdot e_{l,r} \right) \left(\mathrm{mod}\ 2^b - 1 \right) \right\}$$

$$\vdots$$

$$M_k = \left\{ \bigcup_{l=1}^{t} \bigcup_{r=0}^{p-1} \left(-C_k \cdot e_{l,r} \right) \left(\mathrm{mod}\ 2^b - 1 \right) \right\}$$

$$M_{k+1} = \left\{ \bigcup_{l=1}^{t} \bigcup_{r=0}^{p-1} \left(e_{l,r} \right) \left(\mathrm{mod}\ 2^b - 1 \right) \right\}$$

Now, suppose that the coefficients C_i are chosen in such a way that each one multiplied (modulo $2^b - 1$) by each $e_{l,r}$ yields a different result. In that case, it will hold that

$$M_1 \cap \cdots \cap M_k \cap M_{k+1} = \varnothing,$$

$$|M_1| = \cdots = |M_k| = |M_{k+1}|.$$

As a consequence, the set s_1 will have

$$|s_1| = (k+1) \cdot |M_{k+1}| = (k+1) \cdot p \cdot \sum_{l=1}^{t} \binom{d}{l}$$

nonzero elements. In a similar way Condition 2 says that HDBA errors generate $(k+1) \cdot (p-1) \cdot \left[\sum_{n=1}^{t'} \binom{d}{n} \right]^2$ syndromes that are nonzero. To prove this, note that the set s_2 can be expressed as

$$s_2 = \bigcup_{i=1}^{k+1} N_i$$

where

$$N_1 = \left\{ \bigcup_{m=1}^{t'} \bigcup_{n=1}^{t'} \bigcup_{s=0}^{p-2} \left(-C_1 \cdot e_{m,n,s} \right) \left(\mathrm{mod}\ 2^b - 1 \right) \right\}$$

$$\vdots$$

$$N_k = \left\{ \bigcup_{m=1}^{t'} \bigcup_{n=1}^{t'} \bigcup_{s=0}^{p-2} \left(-C_k \cdot e_{m,n,s} \right) \left(\mathrm{mod}\ 2^b - 1 \right) \right\}$$

$$N_{k+1} = \left\{ \bigcup_{m=1}^{t'} \bigcup_{n=1}^{t'} \bigcup_{s=0}^{p-2} \left(e_{m,n,s} \right) \left(\mathrm{mod}\ 2^b - 1 \right) \right\}$$

Given this, suppose that the C_i's are chosen such that

$$N_1 \cap \cdots \cap N_k \cap N_{k+1} = \varnothing$$

$$|N_1| = \cdots = |N_k| = |N_{k+1}|.$$

In that case, it is clear that the set s_2 will have

$$|s_2| = (k+1) \cdot |N_{k+1}| = (k+1) \cdot (p-1) \cdot \left[\sum_{n=1}^{t'} \binom{d}{n} \right]^2$$

nonzero elements. Finally, Condition 3 is a necessary condition for distinguishing LDBA errors from HDBA errors. Therefore, the codes that satisfy the above conditions are $(kb + b, kb)$ integer $(S_{t/d}\text{AEC-DA}_{t'/d}\text{AEC})_b$ codes. □

Theorem 2. *Let ξ be the error set for (kb + b, kb) integer ($S_{t'/d}$AEC-DA$_{t'/d}$AEC)$_b$ codes. Then,*

$$|\xi| = |s_1| + |s_2| = (k+1)\cdot\left[p\cdot\sum_{l=1}^{t}\binom{d}{l} + (p-1)\cdot\left[\sum_{n=1}^{t'}\binom{d}{n}\right]^2\right].$$

Proof. This theorem follows from Theorem 1. □

From Theorems 1 and 2 it is easy to see that the elements of ξ cannot be generated without using a computer. Thus, it is clear that for some particular values of *t'*, *t*, *d* and *b* we cannot know a priori the number and the values of coefficients C_i. In this letter, we have restricted ourselves to practical codes, i.e. to codes with parameters *t'* = 3, *t* = 4, *d* = 8, *b* = 32 and $k \le 64$. The results of the corresponding computer search are given in Table 2.

TABLE II
FIRST 64 COEFFICIENTS FOR INTEGER ($S_{4/8}$AEC-DA$_{3/8}$AEC)$_{32}$ CODES.

2	127	255	511	767	967	1007	1019
1087	1151	1279	1567	1663	1727	1747	1927
1999	2011	2029	2047	2447	2503	2539	2549
2557	2591	2623	2687	2741	2813	2879	2887
3023	3061	3063	3067	3071	3229	3253	3257
3271	3301	3359	3527	3529	3571	3581	3583
3623	3631	3733	3834	3847	3851	3853	4007
4019	4073	4091	4159	4222	4247	4479	4567

C. Error Correction Procedure

From Theorem 2 we know that LDBA and HDBA errors generate $|\xi|$ nonzero syndromes. In addition, from the same theorem, we implicitly know that the relationship between the nonzero syndrome (element of the set ξ), error location (*i*) and error vector (*e*) can be described using (4)-(5). Both these facts imply that the syndrome table requires $|\xi| \times \left[2\cdot b + \lceil log_2(k+1)\rceil\right]$ bits (Fig. 2) to store the error correction data.

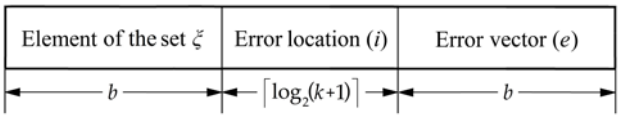

Fig. 2. Bit-width of one syndrome table entry.

Given this, suppose that that the data are received in error ($S \neq 0$). In that case, the decoder will first search the syndrome table to find the appropriate entry. After that, it will execute one of the following operations:

- *for LDBA errors within the i-th data byte*

$$B_i = [\hat{B}_i + e] \pmod{2^b - 1},\ 1 \le i \le k;$$
$$e = [e_{l,r}] \pmod{2^b - 1},\ 1 \le l \le t,\ 0 \le r \le p-1; \quad (6)$$

- *for LDBA errors within the check-byte*

$$C_B = [\hat{C}_B + e] \pmod{2^b - 1};$$
$$e = [e_{l,r}] \pmod{2^b - 1},\ 1 \le l \le t,\ 0 \le r \le p-1; \quad (7)$$

- *for HDBA errors within the i-th data byte*

$$B_i = [\hat{B}_i + e] \pmod{2^b - 1}, 1 \le i \le k;$$
$$e = [e_{m,n,s}] \pmod{2^b - 1}, 1 \le m,n \le t', 0 \le s \le p-2; \quad (8)$$

- *for HDBA errors within the i-th check-byte*

$$C_B = [\hat{C}_B + e] \pmod{2^b - 1};$$
$$e = [e_{m,n,s}] \pmod{2^b - 1}, 1 \le m,n \le t', 0 \le s \le p-2; \quad (9)$$

From the above it is clear that the efficiency of the error correction procedure (in terms of processing time) depends on the number of table lookups. For this reason it is desirable that the elements of ξ are sorted in increasing order. In that case it is possible to use binary search algorithm, which requires n_{TL} table lookups ($1 \le n_{TL} \le \lfloor log_2|\xi| \rfloor + 2$) [12].

Example 1. Let *b* = 8, *d* = 4, *t* = 3, *t'* = 1, *k* = 1 and C_1 = 2. According to Theorem 2, the syndrome table will have $|\xi|$ = 88 entries. Given this, let us assume that we want to transmit 8 bits of data, D = 10110011. In that case, after calculating the value of check-byte C_B

$$C_B = [C_1 \cdot B_1] \pmod{2^b - 1} = [2 \cdot 179] \pmod{255} = 103$$

the codeword C_W = 10110011 01100111 will have 16 bits. Now, let us analyze the following scenarios.

Scenario 1: Suppose that during data transmission an error on the 1st, 3rd and 4th bit has occurred ($\hat{C}_W$ = 00000011 01100111). In that case, the decoder will calculate

$$C_{\hat{B}} = [C_1 \cdot \hat{B}_1] \pmod{2^b - 1} = [2 \cdot 3] \pmod{255} = 6$$
$$S = [C_{\hat{B}} - \hat{C}_B] \pmod{2^b - 1} = [6 - 103] \pmod{255} = 158$$

in order to check whether the value *S* = 158 belongs to the set ξ (Table 3). After completing this task, it will perform error correction by using

$$B_1 = [\hat{B}_1 + e] \pmod{2^b - 1} = [3 + 176] \pmod{255} = 179.$$

Scenario 2: Let us assume that during data transmission an error on the 10th and 16th bit has occurred ($\hat{C}_W$ = 10110011 00100110). Similar to the previous case, after calculating

$$C_{\hat{B}} = [C_1 \cdot \hat{B}_1] \pmod{2^b - 1} = [2 \cdot 179] \pmod{255} = 103$$
$$S = [C_{\hat{B}} - \hat{C}_B] \pmod{2^b - 1} = [103 - 38] \pmod{255} = 65$$

the decoder will conclude that the value *S* = 65 indicates an error within the check-byte (Table 3). As a consequence, the following procedure will take place:

$$C_B = [\hat{C}_B + e] \pmod{2^b - 1} = [38 + 65] \pmod{255} = 103.$$

III. IMPLEMENTATION STRATEGY

From (1)-(3) and (6)-(9) it is clear that the encoder/decoder uses integer and lookup table (LUT) operations. Since these operations are supported by all processors, it is interesting to discuss how the proposed codes can be implemented on modern architectures. Without loss of generality, we will restrict ourselves to eight-core processors (Fig. 3) and integer ($S_{4/8}$AEC-DA$_{3/8}$AEC)$_{32}$ codes.

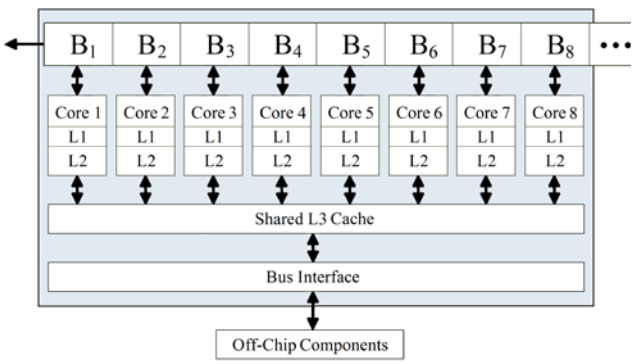

Fig. 3. Block diagram of eight-core processor.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2016.2644661, IEEE Communications Letters

TABLE III
THE SYNDROME TABLE (LUT_2) FOR (16, 8) INTEGER $(S_{3/4}AEC\text{-}DA_{1/4}AEC)_8$ DECODER.

	Element of the set ξ	i	e		*Element of the set* ξ	i	e		*Element of the set* ξ	i	e		*Element of the set* ξ	i	e
1	1	2	1	**23**	34	2	34	**45**	128	2	128	**67**	222	1	144
2	2	2	2	**24**	36	2	36	**46**	129	2	129	**68**	223	1	16
3	3	2	3	**25**	40	2	40	**47**	130	2	130	**69**	224	2	224
4	4	2	4	**26**	48	2	48	**48**	132	2	132	**70**	227	1	14
5	5	2	5	**27**	62	1	224	**49**	136	2	136	**71**	229	1	13
6	6	2	6	**28**	63	1	96	**50**	144	2	144	**72**	231	1	12
7	7	2	7	**29**	64	2	64	**51**	158	1	176	**73**	233	1	11
8	8	2	8	**30**	65	2	65	**52**	159	1	48	**74**	235	1	10
9	9	2	9	**31**	66	2	66	**53**	160	2	160	**75**	237	1	9
10	10	2	10	**32**	68	2	68	**54**	175	1	40	**76**	238	1	136
11	11	2	11	**33**	72	2	72	**55**	176	2	176	**77**	239	1	8
12	12	2	12	**34**	80	2	80	**56**	183	1	36	**78**	241	1	7
13	13	2	13	**35**	94	1	208	**57**	187	1	34	**79**	243	1	6
14	14	2	14	**36**	95	1	80	**58**	189	1	33	**80**	245	1	5
15	16	2	16	**37**	96	2	96	**59**	190	1	160	**81**	246	1	132
16	17	2	17	**38**	111	1	72	**60**	191	1	32	**82**	247	1	4
17	18	2	18	**39**	112	2	112	**61**	192	2	192	**83**	249	1	3
18	20	2	20	**40**	119	1	68	**62**	207	1	24	**84**	250	1	130
19	24	2	24	**41**	123	1	66	**63**	208	2	208	**85**	251	1	2
20	31	1	112	**42**	125	1	65	**64**	215	1	20	**86**	252	1	129
21	32	2	32	**43**	126	1	192	**65**	219	1	18	**87**	253	1	1
22	33	2	33	**44**	127	1	64	**66**	221	1	17	**88**	254	1	128

1) *L1 caches.* The L1 cache is the smallest and fastest type of cache. Its size is limited to 64 KB (per core), whereas the data can be accessed in 1-5 clock cycles [8]. Both these facts suggest that the L1 can be used for storing the coefficient table (LUT_1) whose size is $k \times b$ bits (Table 4).

2) *L2 caches.* The L2 cache is somewhat slower and larger than the L1. Precisely, its size is limited to 512 KB (per core), whereas the individual entries can be accessed in 8-15 clock cycles [8]. Based on these parameters, it is clear that the L2 cannot be used for storing the syndrome table (LUT_2).

3) *L3 cache.* Unlike L1 and L2 caches, the L3 is shared among all cores. Due to this reason, this cache has the highest access latency (25-50 clock cycles). On the other hand, its size is sufficiently large (20-32 MB [8]) to store any LUT_2.

4) *Processing cores.* From [7], [8] it is known that each core has at least one unit that performs 32/64-bit integer operations. Thus, it is clear that the encoding/decoding algorithm can be parallelized using eight threads. The similar applies for the error correction procedure. For instance, *Core* 1 can be used for performing binary search over first $|\xi|/8$ entries, *Core* 2 for performing binary search over next $|\xi|/8$ entries, and so on.

TABLE IV
LOOK-UP TABLE SIZES FOR SOME INTEGER $(S_{4/8}AEC\text{-}DA_{3/8}AEC)_{32}$ CODES.

Code	Encoder	Decoder		
	LUT_1	LUT_1	LUT_2	
	Size	Size	Size	# of Table Lookups
(512, 480)	60 B	60 B	3.54 MB	$1 \le n_{TL} \le 20$
(544, 512)	64 B	64 B	3.82 MB	$1 \le n_{TL} \le 20$
(1024, 992)	124 B	124 B	7.19 MB	$1 \le n_{TL} \le 21$
(1056, 1024)	128 B	128 B	7.52 MB	$1 \le n_{TL} \le 21$
(2048, 2016)	252 B	252 B	14.58 MB	$1 \le n_{TL} \le 22$
(2080, 2048)	256 B	256 B	15.02 MB	$1 \le n_{TL} \le 22$

IV. CONCLUSION

In this letter, we proposed a class of integer codes capable of correcting high-density asymmetric errors within a *b*-bit byte. We have shown that these codes can be implemented "for free", i.e. without modifying the network hardware. Thanks to this feature, the proposed codes have high potential to be used in practice. This primarily refers to optical networks without optical amplifiers in which all nodes possess powerful processors.

REFERENCES

[1] R. Ramaswani, K. Sivarajan and G. Sasaki, *Optical Networks: A Practical Perspective*, 3rd ed., Elsevier, Inc., 2010.

[2] J. R. Pierce, "Optical Channels: Practical Limits with Photon Counting," *IEEE Trans. Communications*, vol. 26, no. 12, pp. 1819-1821, Dec. 1978.

[3] P. Oprisan and B. Bose, "ARQ in Optical Networks," *Proc. IEEE Int'l Symp. Pacific Rim Dependable Computing*, pp. 251-257, Dec. 2001.

[4] CCITT Study Group XVIII Contribution D21, "Observations of Error Characteristics of Fiber Optic Transmission Systems," Jan. 1989.

[5] D. Mello, E. Offer and J. Reichert, "Error Arrival Statistics for FEC Design in Four-Wave Mixing Limited Systems,"*Proc. Optical Fiber Communication Conference*, pp. 529-530, Mar. 2003.

[6] L. James, "Error Behaviour in Optical Networks", PhD thesis, Dept. of Engineering, University of Cambridge, 2005.

[7] R. Giladi, *Network Processors: Architecture, Programming, and Implementation*, Elsevier, Inc., 2008.

[8] L. Johnsson, "Introduction to HPC architecture," Dept. Computer Sciences, Univ. Houston, Houston, TX, USA, Jan. 2014.

[9] A. Radonjic and V. Vujicic, "Integer Codes Correcting Burst Errors within a Byte," *IEEE Trans. Computers*, vol. 62, no. 2, pp. 411-415, Feb. 2013.

[10] A. Radonjic, K. Bala and V. Vujicic, "Integer Codes Correcting Double Asymmetric Errors," *IET Communications*, vol. 10, no. 14, pp. 1691-1696, Sep. 2016.

[11] A. Radonjic and V. Vujicic, "Integer Codes Correcting Spotty Byte Asymmetric Errors," *IEEE Comm. Letters*, vol. 20, no. 12, pp. 2338-2341, Dec. 2016.

[12] K. Mehlhornand and P. Sanders, *Algorithms and Data Structures: The Basic Toolbox*, Springer, 2008.