International Journal of Engineering & Technology, 7 (3) (2018) 1008-1013

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet.v7i3.12681
Research paper

FPGA-based redundancy bits reduction algorithm using the enhanced error detection correction code

Lean Karlo S. Tolentino [1] *, Maria Victoria C. Padilla [1], Ronnie O. Serfa Juan [1]

[1] *Department of Electronics Engineering, Technological University of the Philippines, Manila, Philippines*
**Corresponding author E-mail: leankarlo_tolentino@tup.edu.ph*

Abstract

To ensure an error-free transmission in packet switching, additional check bits (either header or a payload) are typically appended to the input data of a message for error detection especially in a string of binary code. Normally, it comes from the input message and as a result of a deterministic algorithm after these data have been processed. The receiver system implements the said algorithm, while the transmitter used it to match the reliability of the sent information and detects whether an error bit has occurred or not. The corrupted bits will be corrected, recovered, and matched with the original message. To further improve the detection and correction of the corrupted transmitted bits, an enhanced error detection correction code implementation was proposed and developed in this paper. This will improve the limitations of using cyclic redundancy checking (CRC) code and Hamming code, by reducing the number of the redundancy bits 'r' in CRC due to the needed polynomial generator, and the overhead of interspersing of the r in conventional Hamming code, respectively. Xilinx Spartan 6 (XC7Z020-2CLG4841) FPGA was used to synthesize the proposed enhanced error detection code (EEDC) method. Based on the results, the transmission rate is faster, and an increase in detection of random errors compared with using CRC and Hamming codes.

Keywords: *Cyclic Redundancy Check; Data Communication; Error Correction; Field Programmable Gate Arrays; Parity Check Codes*.

1. Introduction

A delay of any data transmission system can be affected by many factors as presented in Equation 1. The delay is dependent on the time to propagate the signal across the medium t_{prop} and on the overall sum of bits in a message or packet (L), and R which is the transmission rate of the digital system measured in bps. Also, it is related to the distance between the transmitter and the receiver in meters (d), the propagation speed (c) which is depended on the physical medium of the transmission channel/link (e.g. glass/optical fiber, copper wire, coaxial cable, etc.) with a range of 200,000-300,000 km/s (speed of light) [1], [2] as shown in Figure 1. Also, sending a group of binary codes over the digital network requires redundant bits or extra bits to ensure that no bits were lost during the transmission. However, this process adds the propagation delay of the transmission.

$$Delay = t_{prop} + \frac{L}{R} = \frac{d}{c} + \frac{L}{R} \quad (1)$$

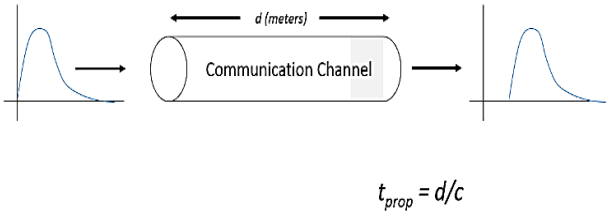

Fig. 1: Transmission Delay in a Communication Channel.

Propagation delay is defined as the length of time taken for the quantity of interest to reach its destination [3]. Existing methods that can be implemented to decrease the propagation delay such as data compression that decrease L and the utilization of higher-speed modems to increase R [2], [4]. Moreover, some procedures of compressing the bit representations of messages consist of a sequence of symbols and involve less bits. In executing these algorithms, the original information must be preserved, and the compressions must be lossless and reversible. Additionally, queuing delay q_{delay} is another factor that needs to be considered in data transmission and computer networking [1], [5 - 8]. At the queue, the packet experiences a queuing delay as it waits to be transmitted onto the link. The queuing delay of a specific packet will depend on the number of other, earlier-arriving packets that are queued and waiting for transmission across the link. The delay of a given packet can vary significantly from packet to packet. The number of bits (L), the average rate (a) at which traffic reaches to the queue measured in packets/sec, the requested bandwidth (B), and the transmission rate R are factors that affect queueing delay [6 - 9]. The said delay is related to the traffic intensity as shown in Equation 2.

$$Traffic\ intensity = \frac{La}{R} = \frac{B}{R} \quad (2)$$

On the other hand, if La/R> 1, the average speed at which bits arrive to the queue exceeds the speed at which the bits can be sent from the queue. Thus, the queue will lean towards increasing and/or approaching infinity. Hence, the system must be designed so that the traffic intensity must not be greater than 1 [1], [7 - 9].
As shown by the two equations (1) and (2), if the total transmitted bits are reduced, a higher transmission speed can be achieved, and the presence of errors or corrupted bits can be lessened. Thus, to ensure a reliable transmission of data, an appropriate error detection

Copyright © 2018 Lean Karlo S. Tolentino et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

and correction method must be implemented. In a typical transmission, an extra parity or check bits are appended to the input message which is derived from this data and undergoes a deterministic algorithm. The same algorithm was used by the receiver to determine and ensure the consistency of the sent information (corrupted or not). The arrived information at the receiver will be recovered and compared so that matching and correction of the corrupted sent bits will be achieved.

Good error detection methods such as the cyclic redundancy checking (CRC) [10] and correction techniques like the Hamming code are being used [11] but these have limitations. To sum up, when implementing CRC codes, there must have a fixed number of the required redundancy bits 'r' due to the allocated polynomial generator which causes the decrease of the transmission speed while when implementing Hamming code, the overhead of interspersing of the computed r is being applied because of the bit's position following the power of binary system. This proposed algorithm is related to the recent work presented in [3], [12 - 14] about CRC and Hamming code. According to these recent researches, polynomial generator of CRC and overhead in Hamming Code increase the propagation delay of the transmission.

In Section 2 and 3 of this paper, the CRC method and Hamming code implementation, respectively, are presented and discussed further. Meanwhile, in Section 4, an alternative error detection method and correction method was proposed and will be named as enhanced-error detection-correction (EEDC) code. The proposed EEDC implementation is an improved form of any current error detection and correction codes. It reduces the limitations of CRC and conventional Hamming code such as the decrease of the transmission rate and the overhead of interspersing of the computed r, respectively. In Section 5, the experimental results were illustrated and the better performance of EEDC over the CRC and Hamming codes in terms of transmission rate and random error detection, respectively, were presented. Lastly, in Section 6, we concluded this paper.

2. Cyclic redundancy checking (CRC) codes

Cyclic Redundancy Checking codes can be referred to as polynomial codes (transmitted bit strings can be interpreted as a polynomial where its coefficients are the bit string values 0 and 1) since every codeword $C(x) = C_{n-1}C_{n-2}\ldots C_0$ can be represented by a polynomial degree n-1 [15], [16] as shown in Equation (3).

$$C(x) = \sum_{i=0}^{n-1} C_i\, x^i \tag{3}$$

In CRC and all existing cyclic codes, each valid code polynomial must be a multiple of a generator polynomial g(x) (in this paper, $g(x) = 10101 = x^4 + x^2 + 1$ for simulation purposes). It shows that this polynomial code as a reference for effective error-correction methods, but it has a fixed number of check bits because it depends on the nth degree of the g(x) that is needed to append during transmission. This, in turn, reduces the transmission rate of the network. Likewise, CRC codes do not enforce correction, it only imposed retransmission every time an error is detected [17].

To illustrate the CRC encoding process, a message polynomial m(x) is created from a specified message e.g. 1010111 using the same method as Eq. (3). Hence, $m(x) = x^6 + x^4 + x^2 + x + 1$.

To illustrate the derivation of the constructed CRC code, the input message m(x) is considered. The input message to be sent for transmission is given as:

$$m_n m_{n-1} m_{n-2} \ldots m_2 m_1 m_0 \tag{4}$$

In terms of the polynomial form:

$$m(x) = m_n x^n + m_{n-1} x^{n-1} + m_{n-2} x^{n-2} + \ldots + m_2 x^2 + m_1 x + m_0 \tag{5}$$

Or

$$m(x) = \sum_{i=0}^{n-1} m_i\, x^i \tag{6}$$

Then, it is multiplied by x^k, which is the degree of the generator polynomial g(x). The resulting product is divided by g(x). The resulting quotient is designated as q(x) and is discarded. The remainder which is denoted as R(x) is the CRC code (0111 in this paper; $x^2 + x + 1$). Hence, it can be said that:

$$C(x) = x^k\, m(x) = q(x)g(x) + R(x) \tag{7}$$

The coefficients of the remainder polynomial R(x) will be treated as parity/check bits and will be attached to the message before sending it for transmission.

Hence, the bits of the message to be transmitted which will be denoted as T(x) is represented by a polynomial:

$$T(x) = x^k\, m(x) + R(x) \tag{8}$$

The decoding process is similar to the encoding step. It separates each word received into the message and the remainder portion, and checks if the computed remainder from the message corresponds to the transmitted bits. An error will happen if there is a mismatch and a retransmission (ARQ) of message will be requested by the receiver.

Although CRC is easy to execute using binary hardware like FPGAs, it is not guaranteed that the intentional alteration of data will not occur. In addition, in CRC, an overflow of data may be expected. Hence, it must have an error correction method besides using CRC for an efficient system. Moreover, part of the drawbacks of CRC is Serial Architecture which consumes more time to send the information [18].

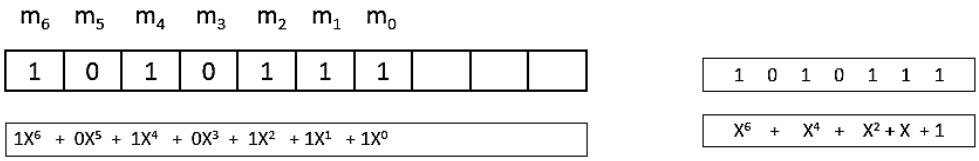

Fig. 2: Polynomial Representation of a Binary Codeword for Message m(x) = 1010111. (a) Binary Pattern and (b) Short Form.

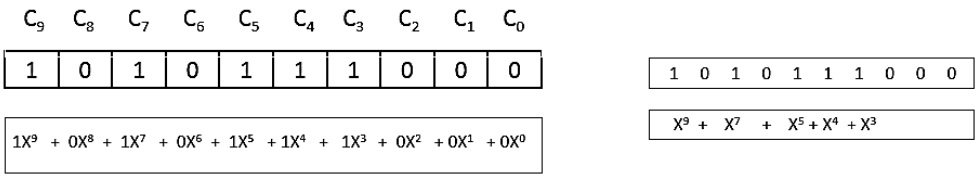

Fig. 3: Polynomial Representation of a Binary Codeword for C(x) = 1010111000. (a) Binary Pattern and (b) Short Form.

Below is the pseudocode of CRC code.

```
   function crc(bit array bitString[1..len], int len)
1      {remainderPolynomial := polynomialForm(bitString[1..n])
2        // First n bits of the message
3        for i from 1 to len
4          {remainderPolynomial := remainderPolynomial * x + bitString
               [i+n] * x0
5        // Define bitString[k]=0 for k>len
6            if coefficient of xn of remainderPolynomial = 1
7            {remainderPolynomial := remainderPolynomial xor
                 generatorPolynomial
8            }
9          }
10       return remainderPolynomial
11     }
```

As shown in the first part of the pseudo-code, the remainder is initialized with leading portion of the input bit sequence. Then, each step, the remainder is up-shifted, and the emptied register is filled with the next bit from the input bit string. And the input bit string needs to be appended with zero. These zeros will flush out the data through the remainder during computation.

3. Hamming codes

Hamming code is one type of error-correcting and linear block codes used to detect and correct transmission error bits. Also, two concurrent error bits can be spotted, and a single error bit can be altered [19 - 24]. As shown in Figure 4, in Hamming code, an n-bit data word (D) are attached to the redundancy/parity bits 'r', creating a single word and having a total number of bits of D + r bits.

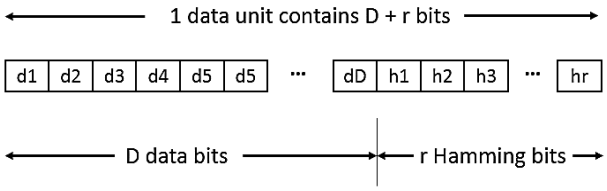

Fig. 4: A Data Unit Comprised of D Character Bits and R Hamming Bits [20].

At least D + r + 1 different codes must be specified by the n Hamming bits since the Hamming bits should be able to recognize what bit is corrupted or in error. An absence of errors must be specified by one code of the D + r codes. Meanwhile, the bit position where there is a presence of error must be recognized by the rest of the D + r codes. Hence, the number of 'r' which is needed is given by the equation below since r bits can generate 2r different codes:

$$2^r \geq D + r + 1 \tag{9}$$

The said redundancy bits are to be placed at binary bit positions behind the original data bits. At that moment, the entire bit positions are allotted for the data to be coded. As a result, an increase in overhead because of interspersing the redundancy bits both for the transmitter and receiver parts.

Table 1 summarizes the data bits position and its corresponding Hamming bits where X value, which denotes a don't care condition, is in a non-sequential or random format.

Moreover, the advantage of using Hamming code is it improves robustness [25] and better in performance on networks where the data streams are susceptible to single-bit errors. However, if multiple errors are present, the Hamming codes can detect the errors, but the resultant could cause another correct bit to be altered, causing an additional error to be present on the data as shown in [26].

Table 1: Data Bits Position and Corresponding Hamming Bits

Bit Position	1	2	3	4	5	6	7	…
Power of 2	2^0	2^1		2^2				…
Encoded position	r_1	r_2	D_1	r_3	D_2	D_3	D_4	…
Required parity bits position for 'r'	r_1		X		X		X	…
		r_2	X			X	X	…
				r_3	X	X	X	…

4. Enhanced error detection correction (EEDC) codes

This proposed algorithm is an enhanced version of CRC and Hamming codes. The concept of this proposed algorithm is based on these two error detection codes. Moreover, to show it is efficient, this was compared with CRC and Hamming codes only. The Enhanced Error Detection Correction (EEDC) codes were proposed and developed from the modified and improved Hamming codes. Figure 6 shows the flowchart of the proposed EEDC algorithm. In this method, each valid codeword of C bits comprises the valid input data bits Di. An invalid codeword can be generated when there are C bits that can be altered in any valid Di entity. Thus, the overall number of codewords conforming to a valid data entity is C + 1. Meanwhile, the overall number of codewords is $(C + 1)2^{Di}$ for every 2^{Di} valid data patterns. The possible number of patterns is 2^C in every Di bit codeword. With this, the number of valid and invalid codes that can exist will be limited. Thus,

$$(C + 1)2^{Di} \leq 2^C \tag{10}$$

And, it can be stated as

$$C = D_i + r \tag{11}$$

And

$$(D_i + r + 1)2^{Di} \leq 2^{Di+r} \tag{12}$$

So, the specified state of the inequality below should be satisfied by the overall number of the needed redundancy bits:

$$(D_i + r + 1) \leq 2^r \tag{13}$$

Both the input data D_i and the needed r will be transformed into a polynomial form with a degree of n-1 as shown by the equation below:

$$D(x) = \sum_{i=0}^{n-1} D_i \, x^i \tag{14}$$

And

$$r(x) = \sum_{i=0}^{n-1} r_i \, x^i \tag{15}$$

Then, the degree of polynomial D(x) will be multiplied by the nth value of r:

$$G(x) = D(x) \bullet X^n \tag{16}$$

Thus, the structure of the EEDC codes is finalized as shown in the equation below:

$$\text{EEDC codes} = G(x) + r(x) \tag{17}$$

Suppose a 7-bit data input of 1001110 which will be implemented by using the EEDC codes. The polynomial form of the 7-bit data input is $X^6 + X^3 + X^2 + X$. Hence, the smallest number of r to meet the above inequality of (13) is 4, and its polynomial form is $r_3X^3 + r_2X^2 + r_1X + r_0$.

Using equation (16), G(x) value is $X^{10} + X^7 + X^6 + X^5$.

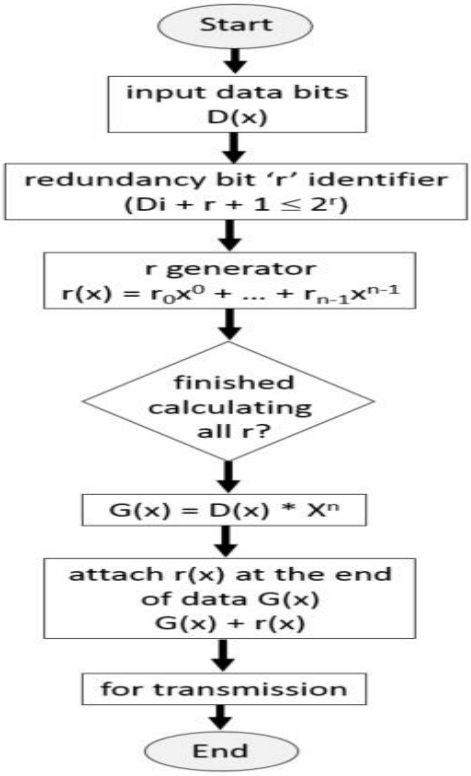

Fig. 5: Flowchart of the EEDC Method.

The check bits in position 8 and 9 are located at bit positions 1, 3, 5, 7, and 2, 3, 6, 7, respectively. Even parity and odd parity is expected for check bits in position 8 and 9, respectively so the value of r_3 is 0 while the value of r_2 is 1.

The check bits in position 10 are located at bit positions 4, 5, 6, 7. Odd parity is expected so r_1 equals 1. On the other hand, the check bits in position 11 are the redundancy bits r_3, r_2, and r_1 only. Even parity is expected so r_0 equals 0.

Finally, the EEDC codes for this 11-bit data which will be sent is $X^{10} + X^7 + X^6 + X^5 + X^2 + X$ in polynomial form while it is $1001110r_3r_2r_1r_0$ in binary form. The redundancy bit values of r_3, r_2, r_1, and r_0 are 0110, respectively, and are to be attached at bit positions 8, 9, 10, and 11, respectively. So, the 11-bit data value is 10011100110.

To evaluate the proposed method, an extensive experimental analysis is conducted. The set-up parameters have been used to determine the performance of the system. Also, this has been implemented in Xilinx Spartan 6 (XC7Z020-2CLG4841) FPGA. Figure 6 shows the EEDC generator part only. The Verilog hardware description language was used since FPGAs are now widely used to reconfigure any embedded system [27 - 30]. Also, most FPGA designs are cost-effective than that of ASIC implementations. Moreover, in designing using FPGA, modifications on software or hardware do not require extensive changes [28 - 31].

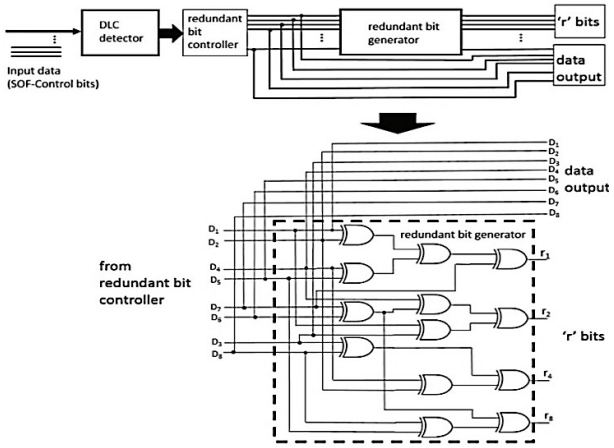

Fig. 6: EEDC Code Generator Architecture.

5. Experimental testing and results

Table 2 shows the results of the simulated data rate comparison between CRC, Hamming codes and the proposed EEDC in an 8-bit frame data. It shows that the transmission rate of this proposed implementations is faster compared with CRC and Hamming Codes.

During testing, the following set-up is needed to satisfy the condition of simulation. The number of bytes' frame is between 1 byte and 8 bytes, random data were feed in designed system including the FPGA board. Figure 7 shows that the EEDC algorithm has better performance in detection of errors unlike in using CRC and Hamming codes. Figure 8 shows the simulation results in using the proposed EEDC, as the data sent as NRZ bytes with a transit start sequence from 1-8 bytes.

Table 2: Data Rate Comparison between CRC, Hamming, and the Proposed EEDC Codes

Transmitted data (bytes)	Data Rate (frames per second)		
	CRC	Hamming	EEDC
1	2,080	2,450	2,630
2	1,785	2,049	2,504
3	1,582	1,785	1,956
4	1,404	1,562	1,600
5	1,275	1,388	1,422
6	1,157	1,250	1,398
7	1,059	1,136	1,344
8	984	1,050	1,119

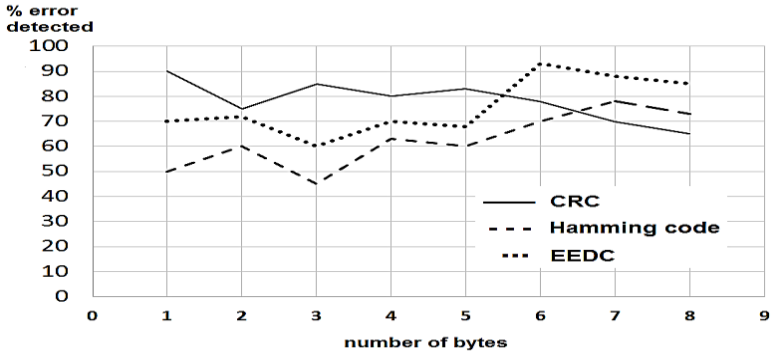

Fig. 7: Error Detection Performance of CRC, Hamming and EEDC Codes.

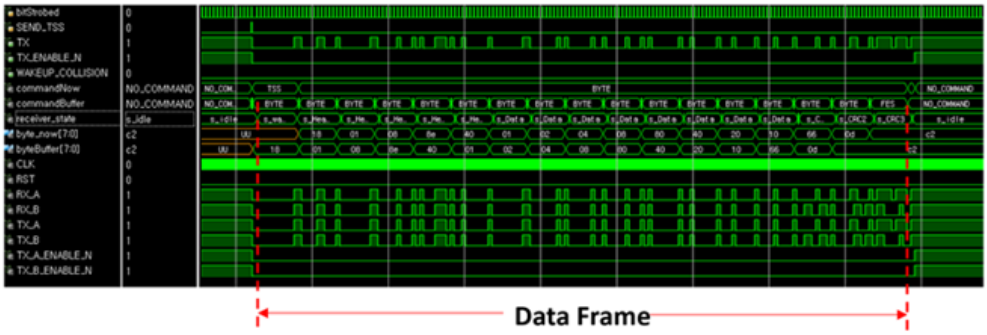

Fig. 8: Simulation Result Using EEDC Codes.

6. Conclusion

In this paper, the proposed enhanced error correction and detection (EEDC) codes can be used as an alternative error detection and correction technique. As shown by the results, using the proposed EEDC, a faster data rate, and a minimized overhead payload were achieved against CRC and Hamming codes whose drawbacks is the fixed data frame of the polynomial generator, and additional usage of overhead payload, respectively.

Acknowledgment

This study is supported by the Technological University of the Philippines through the University Research and Development Services Office, University Research and Extension Council and the College of Engineering.

References

[1] Kurose JF & Ross KW (2012), "Computer networking: a top-down approach", *Pearson*, pp. 36-44.

[2] Leon-Garcia A & Widjaja I (2003), "Communication networks," *McGraw-Hill*, Inc. pp. 101-105.

[3] Baker RP (1998), "Computer assisted survey information collection," Vol. 66, *John Wiley & Sons*, pp. 320-321.

[4] Ullah S, Khan J, Latif S, & Ullah I (2011), "Indication of Efficient Technique for Detection of Check Bits in Hamming Code", *International Journal of Computer Science Issues*, Vol. 8, No. 5, pp. 241-246.

[5] Wu J, Liu H, Dobre OA, Fang C, & Qian L (2013), "CA-MAC: A Novel MAC Protocol to Alleviate Congestion in Wireless Sensor Networks," *Advances in Electrical and Computer Engineering*, Vol. 13, No. 4, pp. 41-46. https://doi.org/10.4316/AECE.2013.04007.

[6] Poudyal N, Lee HC, Kwon YJ, & Lee BS (2011), "Delay-bound Admission Control for Real-time Traffic in Fourth Generation IMT-Advanced Networks based on 802.16m," *Advances in Electrical and Computer Engineering*, Vol. 11, No. 1, pp. 31-38. https://doi.org/10.4316/AECE.2011.01005.

[7] Chen CL, Lai YL, Chen CC, & Chen KC (2009), "Construction of a Real-Time and Secure Mobile Ticket System," Journal of Information Science & Engineering, Vol. 25, No. 3, pp. 807-825.

[8] Beck MT & Linnhoff-Popien C (2014), "On delay-aware embedding of virtual networks," *Proceedings of the AFIN 2014: The sixth international conference on advances in future internet*, Lisbon, Portugal, pp. 55-59.

[9] Venkataram P, Chaudhari S, Rajavelsamy R, Ramamohan TR, & Ramakrishna H (2004), "Disk-oriented VCR operations for a multiuser VOD system," *Journal of the Indian Institute of Science*, Vol. 84, No. 5, pp. 123-140.

[10] Irvin IR (2003), "Cyclic redundancy checks with factorable generators," *IEE Proceedings – Communications*, Vol. 150, No. 1, pp. 17-20. https://doi.org/10.1049/ip-com:20030189.

[11] Park SI & Yang KC (2002), "Extended Hamming accumulate codes and modified irregular repeat accumulate codes," *Electronics Letters*, Vol. 3, No. 10, pp. 467 – 468. https://doi.org/10.1049/el:20020316.

[12] Kora P (2017), "Single Bit Error Detection and Correction using Novel Hamming Code Method", Imperial Journal of Interdisciplinary Research, Vol. 3, No. 1, pp. 1285-1288.

[13] Gad VR, Gad RS, & Naik GM (2015), "Configurable CRC Error Detection Model for Performance Analysis of Polynomial: Case Study for the 32-Bits Ethernet Protocol", *Lecture Notes in Computer Science*, Vol. 9247, pp. 529-542. https://doi.org/10.1007/978-3-319-23126-6_46.

[14] Singh S, Sujana S, Babu I, & Latha K (2013), "VLSI Implementation of Parallel CRC using Pipelining, Unfolding and Retiming", *IOSR Journal of VLSI and Signal Processing (IOSR-JSVP)*, Vol. 2, No. 5, pp. 67-72.

[15] Koopman P & Chakravarty T (2004), "Cyclic redundancy code (CRC) polynomial selection for embedded networks," *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, Florence, Italy, pp. 145-154. https://doi.org/10.1109/DSN.2004.1311885.

[16] Serfa Juan RO & Kim HS (2018), "Implementation of EEDC for Trailer Segment in Enhanced FPGA-based FlexRay Controller," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, Vol. 10, No. 1-9, pp. 161-166.

[17] Serfa Juan RO, Jeong MW, Kim HS (2016). "Development of burst error effect reduction algorithm for CAN using interleaver method," *2016 International SoC Design Conference (ISOCC)*, Jeju, South Korea, pp. 165-166. https://doi.org/10.1109/ISOCC.2016.7799843.

[18] Muthiah D & Raj AAB (2012), "Implementation of high-speed LFSR design with parallel architectures," *Proceedings of the 2012 International Conference on Computing, Communication and Applications (ICCA)*, Tamilnadu, India, pp. 1-6. https://doi.org/10.1109/ICCCA.2012.6179137.

[19] Bertozzi D, Benini L, & De Micheli G (2005), "Error control schemes for on-chip communication links: the energy-reliability tradeoff," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24, No. 6, pp. 818-831. https://doi.org/10.1109/TCAD.2005.847907.

[20] Tomasi W (2014), "Advanced Electronic Communications Systems," 6th edition, *Pearson*, pp. 161-168.

[21] Forouzan BA (2013), "Data communications & networking," *McGraw-Hill*, pp. 257-273.

[22] Noorbasha F & Suresh K (2018). "FPGA implementation of RGB image encryption and decryption using DNA cryptography". *International Journal of Engineering & Technology*, Vol. 7, No. 2.8, pp. 397-403. https://doi.org/10.14419/ijet.v7i2.8.10469.

[23] Nasaruddin N, Yuhanda B, Elizar E, & Syahrial S (2017), "Design and Performance Analysis of Channel Coding Scheme based on Multiplication by Alphabet-9," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, Vol. 9, No. 1, pp. 7-13.

[24] Bulo Y, Anju M, & Bhunia CT (2016), "Implementing (7, 4) Hamming Code with Extra One Parity Bit and Bit Reverse Scheme To Correct Errors At The Receiver Side," *International Journal of Applied Engineering Research*, Vol. 11, No. 1, pp. 22-27.

[25] Bucur IG, Stanescu D, & Stratulat M (2013), "Enhanced Segment Compression Steganographic Algorithm," *Advances in Electrical and Computer Engineering*, Vol. 13, No. 3, pp. 101-106. https://doi.org/10.4316/AECE.2013.03016.

[26] Sanchez-Macian A, Reviriego P, & Maestro JA (2012), "Enhanced detection of double and triple adjacent errors in hamming codes through selective bit placement," *IEEE Transactions on Device and Materials Reliability*, Vol. 12, No. 2, pp. 357-362. https://doi.org/10.1109/TDMR.2012.2186965.

[27] Sedcole P, Blodget B, Becker T, Anderson J, & Lysaght P (2006), "Modular dynamic reconfiguration in Virtex FPGAs," *IEE Proceedings-Computers and Digital Techniques*, Vol. 153, No. 3, pp. 157-164. https://doi.org/10.1049/ip-cdt:20050176.

[28] Serfa Juan RO & Kim HS (2018), "Reconfiguration of an FPGA-Based Time-Triggered FlexRay Network Controller using EEDC,"

Journal of Circuits, Systems, and Computers, Vol. 27, No. 6, pp. 1-11. https://doi.org/10.1142/S0218126618500962.

[29] Brekhov O & Ratnikov M (2014), "Pipelined Error-detecting Codes in FPGA Testing," *Advances in Electrical and Computer Engineering*, Vol. 14, No. 2, pp. 57-62. https://doi.org/10.4316/AECE.2014.02010.

[30] Serfa Juan RO & Kim HS (2016), "Using DSP Algorithms for CRC in a CAN Controller," *IEIE Transactions on Smart Processing & Computing*, Vol. 5, No. 1, pp. 29-34. https://doi.org/10.5573/IEIESPC.2016.5.1.29.

[31] Tolentino LKS & Beleno DMT (2017), "Development of a 3D Disparity Estimation Processing Architecture," *International Journal of Applied Engineering Research*. Vol. 12, No. 19, pp. 8420-8424.