Journal of Engineering Science and Technology Review 12 (2) (2019) 34 - 39

Research Article

JOURNAL OF
**Engineering Science and Technology Review**

www.jestr.org

# Overhead Interspersing of Redundancy Bits Reduction Algorithm by Enhanced Error Detection Correction Code

**Lean Karlo S. Tolentino[1,2,*], Ira C. Valenzuela[1,3] and Ronnie O. Serfa Juan[4]**

*[1]Department of Electronics Engineering, Technological University of the Philippines, Manila, Philippines*
*[2]University Extension Services Office, Technological University of the Philippines, Manila, Philippines*
*[3]University Research and Development Services Office, Technological University of the Philippines, Manila, Philippines*
*[4]Department of Electronic Engineering, Cheongju University, South Korea*

Received 15 March 2018; Accepted 28 March 2019

***Abstract***

Additional check bits, which are commonly attached to the message's input data, are normally used to minimize the error during data transmission. The receiver system implements a checking algorithm to determine if an error was occurred in the received data. This algorithm will correct a corrupted bit and recover the original message. An enhanced error detection correction code was presented to better detect and correct the corrupted conveyed bits. It improves the existing limitations of utilizing cyclic redundancy checking (CRC), Hamming code, and other checksum techniques. Also, it reduced the length of the redundancy bits which exists in CRC, the overhead of interspersing of the redundancy bits in a typical Hamming code, and the system resources such as processor time and bandwidth in checksum techniques. This paper was synthesized and simulated using the Xilinx Spartan 6 (XC7Z020-2CLG4841) FPGA. Results show that the resource utilization of the designed memory architecture using EEDC is lower compared to CRC, Hamming, and Checksum algorithms.

*Keywords:*checksum, cyclic redundancy checking, enhanced error detection correction code, Hamming, redundancy

## 1. Introduction

In today's setting, digital communication plays a vital role in electronic communication world. However, due to this high level of complexity in hardware and software setups, it implies that the system is also extremely susceptible to errors [1][2]. Therefore, fault tolerance is an absolute requirement for most communication systems [3]. In addition, there are factors affecting the quality of transmission such as environmental interference and infrastructure defects. These can cause random bit errors during data transmission. Also, in digital communication, a quality output depends upon its characteristics. These factors activate the delay data communication such as propagation time $t_{prop}$, transmission rate $R$, distance $d$ between the transmitter and receiver, and the required check sum which is inserted to minimize the error of the system. In addition, each error correct codes requires different length of check bits. These correction bits may consume many overhead bits which contributes in the delay and directly affects the communication system. There are numerous techniques that aims to reduce these drawbacks in communication technology (e.g. propagation delay and required number of check bits). However, the existing algorithms always depends on the number of input bits. Thus, it greatly consumes overhead bits in the transmitted packet of data. Some procedures like compressing the messages' bit representations which consist of order of symbols and contain few bits can help to resolve the communication problem but in implementing these algorithms, a preserved original information and the lossless and reversible compressions must be achieved.

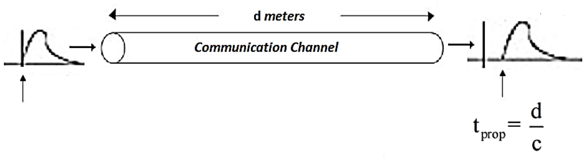


**Fig. 1.** Transmission Delay

Queuing delay is one additional problem which exists in packet transmission. Although it differs from packet to packet, it has a significant impact. As an example, if arrival rate (in bits) is more than the transmission rate of link for a certain period, packet will queue and wait to be transmitted on. Also, some packets can be lost if memory or buffers are filled up. The theoretical concept of queuing delay is show in Equation 1.

$$\text{Traffic intensity} = \frac{L \bullet a}{R} \tag{1}$$

where $L$ is the number of bits, ($a$) as the average rate at which traffic makes it to the queue in packets/s, and $R$ as the transmission rate. If $\frac{L\,(a)}{R} > 1$, the average rate is more than the rate at which the bits can be send. Thus, the queue will bias on the way to approaching infinity. Hence, the system

*E-mail address: leankarlo_tolentino@tup.edu.ph
ISSN: 1791-2377 © 2019 Eastern Macedonia and Thrace Institute of Technology. All rights reserved.
doi:10.25103/jestr.122.05

must be reconfigured to achieve a traffic intensity which is less than 1 [4][5][6][7].

As shown in Equation 1, if the whole transmitted bits are lessened, there will be a higher transmission speed and a low occurrence of errors. Thus, in ensuring a consistent data transfer, a suitable method of detection and correction of error is required. During transmission, there are extra parity (bits for checking) which are attached to the transmitted message. These bits result from their input data and undergone a deterministic algorithm. The said algorithm has been utilized by the receiver to check any errors and confirm the reliability of the transmitted information. The received information will be recovered, compared and matched to achieve the corrected of the sent error bits.

Furthermore, if an error causes in changing a 0 to 1-bit flip in the received data, the information is totally different. Hence, fault tolerance is becoming a matter of concern to provide the ability of a system in maintaining its functionality. A fault tolerant technique allows a system to continue its process properly instead of failing completely. These soft errors can be counteracted by using error correction codes [8][9]. Conventionally, there are two basic techniques in error treatment. First is by adding adequate extra bits, which are relayed into the data stream, in the transmitted data block to allow the receiver to identify what the sent information must have been. Second is to include only adequate redundancy to allow the receiver to determine that error is present, and a request for retransmission is acknowledge. A new method is proposed in this paper which enhances the drawbacks of CRC and Hamming code.

The next section presents and discusses related works in CRC, Hamming codes, and other useful checksum encoding methods. Section 3 presents the proposed different error detection and correction method. In Section 4, the experimental testing and results were presented. Lastly, Section 5 concludes this paper.

## 2. Review of Related Works

There are few existing works which are already implemented in solving the issues in communication errors. A well-known error detection method like the cyclic redundancy checking (CRC) [10][11][12][13] and correction technique such as the Hamming code [14][15][16] are being used but has their own limitations. CRC codes require a constant number of redundancy bits due to the assigned polynomial generator in every application upon implementation. For this reason, the transmission speed decreases. On the other hand, Hamming codes consume overhead bits due to the interspersing of the computed redundancy bit. Another method is called checksum in which it is mostly used to detect errors in data transmission on communication networks. In checksum, the block of data which is being transmitted are added up and its the sum along with the data are transmitted. The received data blocks are added up by the receiver and the matching of the received checksum bits with the calculated checksum are being done [17][18].

### 2.1. Cyclic Redundancy Checking (CRC) Codes

CRCs are applied in most communication networks which deliver low-cost and effective error detection abilities [19]. On transmission mode, as information transmission rates and the amount of the stored data increases, the requirement for an uncomplicated but powerful error detection codes increases. Whenever high-speed transmission rate is essential, serial implementation does not achieve this requirement. However, CRC hardware operation is based on Linear Feedback Shift Registers (LFSRs), which utilizes serial transmission. LFSR is constructed from common shift registers with a few number of XOR gates and is utilized for random number generation and counters.

CRC code can be denoted as polynomial codes (sent strings of bit can be understood as a polynomial wherein its coefficients consist of values of bit string, 0 and 1) since all codewords of the form $C(x) = C_{n-1}C_{n-2}\ldots C_0$ are denoted as a polynomial degree $n$-1 [4] as presented in Equation 2.

$$C(x) = \sum_{i=0}^{n-1} C_i\, x^i \qquad (2)$$

CRC and most cyclic codes demand that "every valid code polynomial be a multiple of a generator polynomial *g(x)*" (for example, g(x) = 10101 = $x^4 + x^2 + 1$; message polynomial m(x) = 1010111). This polynomial code can be referred to as a basis for good error correction methods. However, it contains a constant length of check bits since it rests on the generator polynomial's $n^{th}$ degree that is required to attached throughout the transmission [20]. For this reason, a reduced network transfer rate has been achieved. Similarly, correction is not enforced by CRC codes, retransmission is executed when they encounter an error.

To demonstrate the CRC process, basic bitwise algorithm is used to represent its process. The bitwise algorithm (CRCB) is basically a software implementation utilizing a linear feedback shift register (LFSR). Figure 2 shows a basic hardware implementation. The LFSR is triggered by a clock. In each clock pulse, the input data m(x) is shifted and transmitted into the register. When the entire input bits have been dealt with, the LFSR holds the CRC bits that are shifted out on the data line.

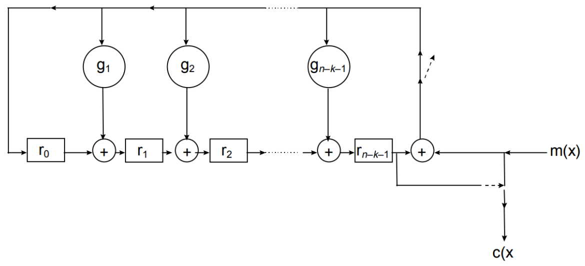


**Fig. 2.** Generating CRC Utilizing a Linear Feedback Shift Register (LFSR)

The following algoritzm can be utilized in the software implementation:

The check bits are assumed to be kept in a CRC register. Then, the implementation using the software resulted to:

1) CRC ← 0.

2) If the CRC left-most bit equals 1, the succeeding bit of the message must be shifted, and the register must be XORed with the generator polynomial; or else, the shift must be done in the succeeding bit of the message.

3) Second step is repeated until the shifting in of all the bits of the improved message were carried out.

There will be a faster implementation by treating the data as higher units than bits when its size is not larger than the generator polynomial's degree. Nevertheless, the speed gain matches to an increase of memory since precomputed values (lookup tables or LUT) will be utilized.

The decoding method is like the encoding process. It splits up every word received into the message and the remainder portion, and it examines if the calculated remainder from the message resembles to the sent bits. An error will be expected if mismatch occurs and the receiver will request for the retransmission (ARQ) of message.

Although, CRC is not difficult to be implemented using hardware such as ASICs and FPGAs. It is not ensured that the intended modification of data will not happen. Likewise, an overflow of data may be likely. Thus, an error correction method must be implemented as well as utilizing CRC for the system to be efficient. Furthermore, a disadvantage of CRC is the Serial Architecture which takes additional time to transmit the message [21].

## 2.2. Hamming Codes

The implementation algorithm using the designed Hamming code encoding and decoding circuit by transmission gate logic is shown in [22]. Simulations has been verified using the tanner tools and results showed a reduced-on channel length and minimized the dissipation on power consumption. The combined Hamming and Hadamard codes have been used in [23] which showed a minimized bound on transmission rate.

In [24], the effectivity of two types of linear block codes namely Hamming and cyclic codes are shown. They are used as error-detecting and error-correcting scheme in long distance communication. The resulting implementation managed to detect and correct errors in a communication channel.

Hamming code is an error-correcting and linear block code which are used for transmitted error bits' detection and correction. Likewise, two simultaneous error bits and a single error bit can be detected and corrected, respectively, using this method [20][25][26][27]. Figure 3 shows that when Hamming code is implemented, the *n*-bit data word (D) is appended to the redundancy bits (r), generating a single word which results to an overall number of bits of *D + r* bits. The required number of Hamming bits n should be at least D + r + 1.

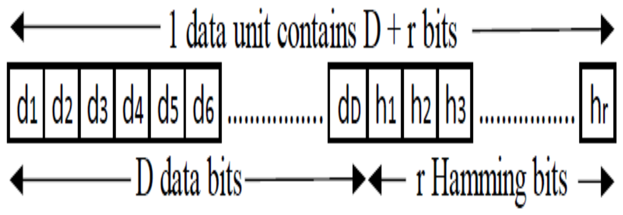


**Fig. 3.** 1 data unit contains D data bits and r Hamming bits [12]

A nonappearance of errors needs to be acknowledged by single code of the D + r codes. Any position of the bit where its location has an occurrence of error needs to be recognized by all the D + r codes. Thus, the amount of required r bits is stated in Equation 3 since $2^r$ unlike codes can be produced by r bits.:

$$2^r \geq D + r + 1 \quad (3)$$

These r bits are to be appended at binary positions of the bit with the unique bits of data. At that point, the whole bit positions are allocated for the data to be coded (i.e., 3, 5, 6, 7, 9, 11, 13, 14, 15, 17, etc.). Therefore, overhead will increase due to interspersing the r bits together for both parts of the transmitter and receiver.

As shown in Table 1, the position of the data bits and the respective Hamming bits are presented where a value of X, which represents a don't care condition, is either in a random or non-sequential form.

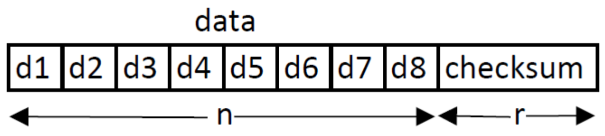
**Table 1.** Position of the Data Bits and Its Respective Hamming Bits

| 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 | … |

Furthermore, Hamming codes performs better on networks where the streams of data are vulnerable to errors of single-bit. Nevertheless, if various errors are existing, these the errors can be detected by Hamming codes. However, it is expected that extra correct bit will be modified and causes an extra error to exist on the data as presented in [28].

## 2.3. Checksum codes

A checksum is a method of checking the redundancy and detecting the errors in a transmission of data in a typical communication network. In a checksum, the entire data block which is being sent are added up and the sum are appended with the data. Then, the received data blocks are added up by the receiver and are checked if these checksum bits correspond to the computed checksum. In the simplest way, a checksum is formed by computing the binary values in a data block using some algorithm and keeping the outputs with the same data. Single-Precision, Double-Precision, Honeywell, and Residue Checksum methods [18][29][30][31][32][33] are the four methods that can be seen in the Checksum Encoder/Decoder.



n = number of bits in the message

r = number of bits before encoding

(n - r) = number of check bits

**Fig. 4.** Checksum Method

In a single-precision checksum, every data byte is added to form a single byte while in a double-precision checksum, every location of the n-bits data is added into the 2n-bits' location. Honeywell checksum is an improved form of double-precision where the entire pairs of consecutive words are appended to form double-precision words. The words are summed in a location whose length is twice the data word size. Lastly, residue checksums are a modified form of single-precision checksums. The carry from the most significant bit (MSB) of the checksum is taken out by the residue checksum. It will be added to the least significant bit (LSB) of the checksum. There are examples where the four checksum methods are implemented [32][33]. The effectiveness of checksums in detecting error during data transmission is presented in [34].

## 3. Enhanced Error Detection Correction (EEDC) Codes

The proposed Enhanced Error Detection Correction (EEDC) code as shown by its flowchart in Figure 5 aims to improve the existing drawbacks of the error detections and corrections above. EEDC demands that every acceptable codeword of *C* bits must contain the acceptable input data bits $D_i$. When the *C* bits are changed in any acceptable $D_i$ unit, an unacceptable codeword will be produced. Therefore, the total number of codewords which corresponds to the acceptable data unit is *C* + 1. For the meantime, the total quantity of codewords is $(C + 1)2^{Di}$ for every $2^{Di}$ acceptable patterns of data. The probable quantity of patterns is $2^C$ in each $D_i$ bit codeword. Hence, a limitation on the number of acceptable and unacceptable codes which may occur will be achieved. Hence,

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

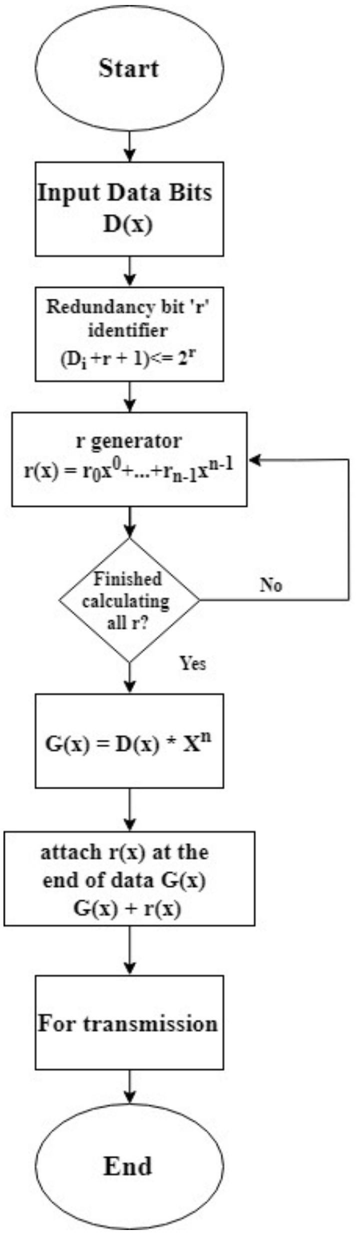

Start
Input Data Bits D(x)
Redundancy bit 'r' identifier $(D_i + r + 1) <= 2^r$
r generator $r(x) = r_0x^0 + ... + r_{n-1}x^{n-1}$
Finished calculating all r?
No
Yes
$G(x) = D(x) * X^n$
attach r(x) at the end of data G(x) G(x) + r(x)
For transmission
End

**Fig 5.** Flowchart of the proposed EEDC algorithm

and, this may be represented by Equation 4

$$C = D_i + r \tag{4}$$

and Equation 5

$$(D_i + r + 1)2^{D_i} \leq 2^{D_i + r} \tag{5}$$

so the inequality shown in Equation 6 should be met by the total number of the desired r bits:

$$(D_i + r + 1) \leq 2^r \tag{6}$$

The input data $D_i$ together with the required r bits will be altered into a polynomial notation with an n-1 degree as presented in Equation 7:

$$D(x) = \sum_{i=0}^{n-1} D_i \, x^i \tag{7}$$

and

$$r(x) = \sum_{i=0}^{n-1} r_i \, x^i \tag{8}$$

The product G(x) consists of the degree of polynomial *D(x)* and the redundancy bit's nth value:

$$G(x) = D(x) \bullet X^n \tag{9}$$

Therefore, the construction of the EEDC code is concluded as presented in Equation 10:

$$\text{EEDC codes} = G(x) + r(x) \tag{10}$$

Assume that an input of 1001110 will be applied by utilizing the EEDC method. The data input in polynomial is $X^6 + X^3 + X^2 + X$. Thus, the least number of *r* to satisfy Equation 9 is 4, and its polynomial result is $r_3X^3 + r_2X^2 + r_1X + r_0$. *G(x)* value is $X^{10} + X^7 + X^6 + X^5$ by applying Equation 16.

To determine the suitable bit of the r bits, $r_3$, $r_2$, $r_1$, and $r_0$ are in position 8, 9, 10 and 11, respectively.

The check bits in 8th and 9th positions are set a 1st, 3rd, 5th, 7th, and 2nd, 3rd, 6th, 7th positions, respectively. Check bits in 8th and 9th positions are applied using even parity and odd parity, respectively, thus the $r_3$ is 0 while $r_2$ equals 1.
The check bit in 10th position is situated at 4th, 5th, 6th, 7th positions. In contrast, the check bits in 11th are only $r_3$, $r_2$, and $r_1$. The value of $r_1$ is 1 and $r_0$ is 0 because odd parity and even parity is encountered, respectively.

Lastly, the EEDC codes in polynomial form and binary form is $X^{10} + X^7 + X^6 + X^5 + X^2 + X$ and 1001110$r_3r_2r_1r_0$, respectively. Bits $r_3$, $r_2$, $r_1$, *and* $r_0$ equals 0110 and are to be appended at 8th, 9th, 10th, and 11th, respectively. Hence, the final EEDC code is 1001110**0110.**

## 4. Experimental Testing and Results

An experimental set-up is conducted to analyze the proposed method against the existing schemes. First, the set-up parameters are necessary to determine the performance of the proposed method. During testing, a set-up is required to meet the condition of simulation. A sequence of frame bytes from 1 byte to 8 bytes was fed as an input to determine the performance of each method.

Table 2 shows the set-up parameters used in the experiment. This was simulated and implemented in Xilinx Spartan 6 (XC7Z020-2CLG4841) FPGA using Verilog hardware description language. FPGAs were commonly chosen to reconfigure embedded devices [35][36] because their implemented designs are cost-effective in contrast with using ASICs and there are no extensive modifications on software or hardware [37][38].

Table 3 shows the resource utilization using the implementation using FPGA. The proposed algorithm (EEDC) was compared with the three existing schemes, namely: CRC, Hamming, and Honeywell checksum. The resource utilization of the designed memory architecture using EEDC is lower compared to CRC, Hamming, and Checksum implementations in terms of the length of look-up tables (LUT), LUTs for implementing distributed RAMs (LUTRAM), utilized flip-flops (FF), and input & output pads (IO). The number of DSP elements for EEDC is lower than CRC and Hamming but same with Checksum implementation. Lastly, global clock buffers were highly utilized in checksum method compared to the other three methods.

**Table 2.** Set-up parameters

| Parameters | Value |
|---|---|
| Number of cycles | 64 |
| Cycle duration | 6 ms |
| Sample clock | 12 ns |
| Payload length | 20 words |

**Table 3**. Resource utilization

| Resources | Hamming | CRC | Checksum | Proposed (EEDC) | Available |
|---|---|---|---|---|---|
| LUT | 7,342 | 7,202 | 8,422 | 6,032 | 54,300 |
| LUTRAM | 756 | 742 | 865 | 726 | 16,400 |
| FF | 5,234 | 4,236 | 5,344 | 4,124 | 108,600 |
| DSP | 3 | 3 | 2 | 2 | 220 |
| IO | 146 | 146 | 168 | 142 | 200 |
| BUFG | 2 | 2 | 3 | 2 | 32 |

## 5. Conclusion

Based on the results obtained, minimized overhead payload bits were achieved using EEDC codes compared to CRC, Hamming code and checksum techniques. The resource utilization of the designed memory architecture using EEDC is the lowest among the CRC, Hamming, and Checksum implementations. The proposed EEDC codes can be used as an alternate error detection and correction technique.

Future work includes implementing the proposed algorithm of enhanced error detection and correction code (EEDC) for other communications applications such as packet transmission to attain a better throughput. Moreover, the developed EEDC implementation will be applied with other overhead implementations.

**Acknowledgements**
This work is funded and supported by the Technological University of the Philippines through the University Research and Extension Council and the University Research and Development Services Office.

## References

[1] J. Yang, P. Wang, Y. Zhang, Y. Cheng, W. Zhao, Y. Chen, and H. H. Li, Radiation-induced soft error analysis of STT-MRAM: A device to circuit approach. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 35, 3, pp. 380–393, (2016).
[2] S. Rehman, K.-H. Chen, F. Kriebel, A. Toma, M. Shafique, J.-J. Chen, and J. Henkel, Cross-layer software dependability on unreliable hardware. IEEE Transactions on Computers 65, 1, pp. 80–94, (2016).
[3] A. Sanchez-Macian, P. Reviriego, and J. A. Maestro, Enhanced detection of double and triple adjacent errors in hamming codes through selective bit placement. IEEE Transactions on Device and Materials Reliability 12, 2, pp. 357–362, (2012).
[4] J. F. Kurose and K. W. Ross, Computer networking: a top-down approach, Pearson (2012).
[5] C.L. Chen, Y.L. Lai, C.C. Chen, and K.C. Chen, Construction of a Real-Time and Secure Mobile Ticket System, Journal of Information Science & Engineering 25, 3, pp. 807-825, (2009)
[6] M. T. Beck and C. Linnhoff-Popien, On delay-aware embedding of virtual networks, AFIN 2014: The sixth international conference on advances in future internet, Lisbon, Portugal, pp. 55-59, (2014).
[7] P. Venkataram, S. Chaudhari, R. Rajavelsamy, T. R. Ramamohan, and H. Ramakrishna, Disk-oriented VCR operations for a multiuser VOD system, Journal of the Indian Institute of Science 84, 5, pp. 123-140, (2004).
[8] X. She, N. Li, and D. W. Jensen, SEU tolerant memory using error correction code. IEEE Transactions on Nuclear Science 59, 1, pp. 205–210, (2012).
[9] M. Imran, Z. Al-Ars, and G. N. Gaydadjiev, Improving soft error correction capability of 4-d parity codes, 14th IEEE European Test Symposium, (2009).
[10] I. R. Irvin, Cyclic redundancy checks with factorable generators, IEE Proceedings – Communications 150, 1, pp. 17-20, (2003).
[11] R. O. Serfa Juan and H. S. Kim, Utilization of DSP algorithms for Cyclic Redundancy Checking (CRC) in Controller Area Network (CAN) controller, 2016 International Conference on Electronics, Information, and Communications (ICEIC), Da Nang, Vietnam, pp. 1-4, (2016).
[12] R. O. Serfa Juan and H. S. Kim, Utilization of High-Speed DSP Algorithms of Cyclic Redundancy Checking (CRC-15) Encoder and Decoder for Controller Area Network, Jurnal Teknologi 78, 5-9, pp. 13-19, (2016).
[13] L. K. S. Tolentino, M. V. C. Padilla, and R. O. Serfa Juan, FPGA-based redundancy bits reduction algorithm using the enhanced error detection correction code, International Journal of Engineering and Technology 7, 3, pp. 1008-1013, (2018).
[14] S. I. Park and K. C. Yang, Extended Hamming accumulate codes and modified irregular repeat accumulate codes, Electronics Letters 3, 10, pp. 467 – 468, (2002).
[15] R. G. Marquart and J. C. Hancock, Performance of Hamming Codes, IEEE Transactions on Space Electronics and Telemetry 9, 4, pp. 115-126, (1963).
[16] N. Shep, and P. H. Bhagat, Implementation of Hamming Code using VLSI, International Journal of Engineering Trends and Technology 4, 2, pp. 186-190, (2013)
[17] Available online: http://www.ecs.umass.edu/ece/koren/FaultTolerantSystems/simulator/Checksum/Checksum_Encoder_Decoder_-_Instructions.html
[18] N. R. Saxena and E. J. McCluskey, Analysis of checksums, extended-precision checksums, and cyclic redundancy checks, IEEE Transactions on Computers 39, 7, pp. 969-975, (1990).

[19] P. Koopman, 32-bit Cyclic Redundancy Codes for Internet Applications, Proceedings International Conference on Dependable Systems and Networks, Washington, DC, USA, pp. 459-468, (2002).

[20] S. Sreelatha and G. Murali, Error correction and detection techniques in quantum cryptography protocol, 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, India, pp. 3584-3588, (2017).

[21] D. Muthiah and A. A. B. Raj, Implementation of high-speed LFSR design with parallel architectures, 2012 International Conference on Computing, Communication and Applications (ICCA), Tamilnadu, India, pp. 1-6, (2012).

[22] D. R. Choudhury and K. Podder, Design of Hamming Code Encoding and Decoding Circuit Using Transmission Gate Logic, International Research Journal of Engineering and Technology 2, 7, pp. 1165-1169, (2015).

[23] Z. Li, S. J. Lin, and H. Hu, On the Arithmetic Complexities of Hamming Codes and Hadamard Codes, Journal of Latex Class Files 14, 8, pp. 1-22, (2017).

[24] I. N. John, P. W. Kamaku, D.K. Macharia, and N. M. Mutua, Error Detection and Correction Using Hamming and Cyclic Codes in a Communication Channel, Pure and Applied Mathematics Journal 5, 6, pp. 220-231, (2016).

[25] B. K. Gupta and R. L. Dua, 30 bit Hamming code for error detection and correction with even parity and odd parity check method by using VHDL, International Journal of Computer Applications 35, 13, pp. 31 – 38, (2011).

[26] W. Tomasi, Advanced Electronic Communications Systems, 6th edition, Pearson, (2014).

[27] J. C. Dimayuga, I. C. Fernandez, A. E. Lopez, R. Pangilinan, L. Alarcon, M. T. de Leon, R. J. Maestro, M. Rosales, and C. V. Densing, A study on the effects of dynamic voltage and frequency scaling on an error detection block for a LoRa communications system, TENCON 2017-2017 IEEE Region 10 Conference, Penang, Malaysia, pp. 1538-1543, (2017).

[28] M. S. Sadi, M. F. Hossain, and M. I R. Shuvo, Tolerating Double Bit Errors by Rearranging Bit Positions, Journal of Modern Computer Networks 1, 2, pp. 1-5, (2017).

[29] R. A. C. Varma and Y. V. Apparao, High-Throughput VLSI Architectures for CRC-16 Computation in VLSI Signal Processing, Lecture Notes in Electrical Engineering 471, pp. 23-32, (2018).

[30] C. E. Stroud, Merging BIST and Concurrent Fault Detection, A Designer's Guide to Built-In Self-Test, pp.267-286, (2002).

[31] A. Azahari, R. Alsaqour, M. Uddin, and M. Al-Hubaishi, Review of error detection of data link layer in computer network, ARPN Journal of Engineering and Applied Sciences 9, 1, pp.1-4, (2014).

[32] B. Peterson, Data Coding and Error Checking Techniques, Virtium Technology, (2015).

[33] I. A. Khan, N. Y. Yun, S. Muminov, and S. H. Park, 2012. A Reliable Error Detection Mechanism in Underwater Acoustic Sensor Networks, Advanced Methods, Techniques, and Applications in Modeling and Simulation, pp. 190-199, (2012).

[34] T. C. Maxino, and P. J. Koopman, The effectiveness of checksums for embedded control networks, IEEE Transactions on Dependable and Secure Computing 6, 1, pp. 59-72, (2009).

[35] A. R. M. Khan, S. M. Gulhane, and S. L. Badjate, FPGA based design & implementation of embedded system for tilt measurement, International journal of advancements in technology 2, 3, pp. 335-349, (2011).

[36] L. Pyrgas, A. Kalantzopoulos, and E. Zigouris, Design and Implementation of an Open Image Processing System based on NIOS II and Altera DE2-70 Board, Journal of Engineering Science and Technology Review 9, 5, (2016).

[37] R. O. Serfa Juan, and H. S. Kim, Reconfiguration of an FPGA-Based Time-Triggered FlexRay Network Controller using EEDC, Journal of Circuits, Systems, and Computers 27, 6, pp. 1-11, (2018).

[38] S. P. Pouros, V. D. Vassios, and D. K. Papakostas, FPGA-Based Mixed-Signal Circuits Testing System Implementation, Journal of Engineering Science and Technology Review 9, 6, pp. 131-134, (2016).