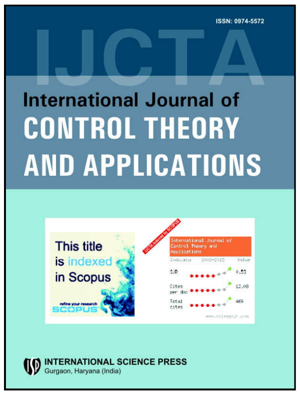

International Journal of Control Theory and Applications

ISSN : 0974-5572

© International Science Press

Volume 10 • Number 6 • 2017

Development of FPGA Based Enhanced Error Detection-Correction Code for CAN Communication

Ronnie O. Serfa Juan[1], Arnold N. Santos[2] and Hi Seok Kim[3]

1 Department of Electronic Engineering Cheongju University, Cheongju City, South Korea and Technological University of the Philippines, Manila, Philippines, Email: ronnieserfajuan@cju.ac.kr
2 Department of Computer Science Modern College of Business and Science, Muscat, Oman, Email: arnoldsantos@mcbs.edu.om
3 Department of Electronic Engineering Cheongju University, Cheongju City, South Korea, Email: khs8391@cju.ac.kr

Abstract: Controller Area Network (CAN) protocol uses cyclic redundancy check (CRC) code for error detections. The main objective of this paper is to use an alternative error detection scheme with error correction implementation called the enhanced error-detection correction code for CAN controller unit. It also aims to reduce the fixed number of required redundancy bits r in CRC due to the required polynomial generator and to possibly increase the CAN's frame rate. The proposed algorithm will position the computed r right after the input bits, instead of placing it in the power of 2 bit position similar to in Hamming code method. This enhanced error-detection correction code eliminates the overhead of interspersing of the required r. The experimental results were synthesized using Xilinx Virtex 5 FPGA and showed a gain in CAN's frame rate. The results also minimized both the overhead payload of interspersing the computed r and the required bit stuff. It also revealed an increase in the detection of random errors. Therefore, this proposed algorithm can be a better option for error detection and correction.

Keywords: CRC, error detection, error correction, enhanced error detection-correction code, Hamming code

1. INTRODUCTION

Several factors affect the transmission delay of any data system. Equation 1 shows these factors for the time to propagate the signal across the medium t_{prop}. The total number of bits in message as L, the speed of the digital transmission system R and measured in bps, d and c are the distance measured in meters and speed of light, respectively.

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

Likewise, there are also several techniques that can be done to reduce this propagation delay, such as data compression that reduce L [1–5] and the usage of higher speed modulation-demodulation to increase R [6].

Some techniques used to compress the bit representations of similar symbols or any messages consist of sequence of equivalent symbols and require a fewer bits. Doing these algorithms require a preservation of all the original information while the compression should be lossless or reversible. Moreover, an error-free channel is required when data compression is used.

The above equation also affects the Controller Area Network (CAN) transmission rate because it uses cyclic redundancy checking (CRC) code [7]. CRC code implementation has a fixed number of required redundancy bits r because of the assigned polynomial generator causing reduction of the transmission rate. A high fault tolerance is required in real time applications of CAN. A demand in high rate proportionally deals with complexity particularly in automotive industry, such as the Advanced Driver Assistance Systems (ADAS) that is designed for road traffic safety to reduce the fatalities in road accidents. Therefore, reducing the total transmitted bits can lead to a higher transmission rate and minimal occurrences of errors or corrupted bits.

Furthermore, a good error detection and correction scheme is required to secure a good error detection and correction method, which CRC does not have because it only implements automatic repeat request (ARQ) [8] when a data mismatch occurs during the detection process. A system without an error correction mechanism cannot fit into an efficient system. Remarkable codes with error correction implementation like the Hamming code exists [9]; however, the overhead payload of interspersing the computed r is used in Hamming code because the bit position follows the power of two-bit system (2^0, 2^1, 2^2…).

Aside from CRC code limitation, CAN system itself has some disadvantages, particularly the limitation in network length that follows the theory in (*1*), which is around 120 feet at 1-Mbps baud rate. Table 1 shows the relationship between baud rates and CAN network lengths. The increase of data frame also contributes in the decrease of the CAN's frame rate [10].

Table 1
Baud rate vs. CAN network length

Baud Rate	*Bit Time*	*Maximum Bus Length*
1 Mbps	1 μsec	25 m
800 kbps	1.25 μsec	50 m
500 kbps	2 μsec	100 m
250 kbps	4 μsec	250 m
125 kbps	8 μsec	500 m
50 kbps	20 μsec	1 km
20 kbps	50 μsec	2.5 km
10 kbps	100 μsec	5 km

Embedding the proposed algorithm for CAN controller system inside an FPGA also has many advantages, such as 100% embedded capability without any required external component. Similarly, this proposal also represents an advancement in the state of the art related to the application of FPGAs in vehicular systems. It further improves power consumption and savings in other factors, such as size and weight. Moreover, FPGAs enable systems to be built with guaranteed deterministic results, which is particularly significant for safety – critical in-vehicle systems, where reliability is primarily important.

The rest of this paper is organized as follows. Section II gives a brief introduction of CAN protocol and its error management methods described in the literature and the related work in this area and it covers the CRC and Hamming code implementations. Section III describes the proposed algorithm as an alternative error detection-correction code and its implementations. Section IV shows the experimental results and interpretation, and Section V discusses the conclusion of the study.

2. CAN PROTOCOL AND RELATED WORK

2.1. CAN Protocol

CAN is a serial communications bus used in various industrial applications. It is a multimaster message broadcast system that specifies a maximum signaling rate of 1 Mbps. The CAN bus specification calls for high immunity to electrical interference and ability to self-diagnose for data errors. This communication protocol is a carrier sense multiple access (CSMA) protocol with collision detection and arbitration on message priority. CSMA causes each node on a bus to wait for a prescribed period of inactivity before attempting to send a message. It helps avoid collisions through a bit-wise arbitration based on a preprogrammed priority of each message in the identifier field of a message. The higher priority identifier always wins bus access. The standard CAN frame, as seen in Fig. 1, shows the standard 11-bit identifier that provides the signaling rates from 125 kbps to 1 Mbps.

		Arbitration Field		Control Field				CRC		ACK			
	SOF	**Identifier**	**RTR**	**IDE**	**r0**	**DLC**	**Data Field**	**Sequence**	**Delimiter**	**slot**	**Delimiter**	**EOF**	**IFS**
Number of bits	1	11	1	1	1	4	up to 64 bits	15	1	1	1	7	3
Assigned bits	0	XXX...XXX	0	0	0	XXXX	XXX...XXX	XXX...XXX	1	1	1	11...	111

Legend for Assigned bits:

1 – Recessive bit	**SOF** – Start of Frame	**CRC** – Cyclic Redundancy Check
0 – Dominant bit	**RTR** – Remote Transmission Request	**ACK** - Acknowledge
X – bit on either state	**IDE** – Identifier Extension	**EOF** – End of Frame
	r0 – Reserved bit	**IFS** – Inter Frame Space
	DLC – Data Length Code	

Figure 1: Standard CAN Data Frame (2.0 A Frame)

2.2. CAN Error Management

Error management in the CAN protocol plays an important part in evaluating the performance of the CAN bus. Every node connected to CAN bus has the capability of detecting an error within the frame. The node that detected the error raises an error flag, halting the transmission on the bus [7]. When a corruption is detected, it immediately signal for retransmission automatically, and based according to priority. Five error detection techniques for error detection are defined in CAN protocol. Below are the different mechanisms used by CAN [11].

A. Bit Monitoring

This transmitting unit monitors the CAN bus and detects an error. Whenever the bit level actually read differs from the transmitted bit, a bit error is signaled.

B. Bit Stuffing

Whenever a CAN transmitter detects five consecutive identical bits in the bit stream, it automatically inserts a complementary bit at the sixth bit position before the bit stream is transmitted. The receivers will remove this extra bit, and if more than five consecutive bits of the same level occurs on the bus, a Stuff Error is signaled.

C. Frame Check

Some parts of the CAN message have a fixed format, such as the CRC Delimiter, ACK Delimiter, End of Frame (EOF), and interframe space (IFS) bits. These frames follow special error checking rules. If a CAN controller detects an invalid value in one of these fixed fields, a Form Error is signaled.

D. Acknowledgment Check

An acknowledgment error error is detected by a transmitter whenever it does not monitor a dominant (logic 0) bit at the Acknowledge Slot [12]. If this bit is recessive (logic 1), it means that none of the nodes received the message properly. Then, an Acknowledgement Error is signaled.

E. Cyclic Redundancy Check

This 15-bit CRC segment in a data or remote frame contains the frame check sequence from start of frame SOF to data field [13]. A CRC error is detected if the result is not the same CRC sequence. This group of bits or remainder is called a syndrome [14].

2.3. CRC and Hamming Code Implementation

A. CRC Code

As shown in Fig. 1, the CRC sequence frame occupies fixed 15 bits because of the required polynomial generator. Minimizing this CRC frame might help to increase the transmission rate. CRC code is an example of a polynomial code referring to the corresponding polynomial of a codeword [15]; thus, the CRC generator used in CAN is CRC-15 with a generating polynomial of $X^{15} + X^{14} + X^{10} + X^{8} + X^{7} + X^{4} + X^{3} + 1$. The idea is to represent every codeword $C(x) = C_{n-1}C_{n-2}\ldots C_0$ as a polynomial of degree n-1. That is, we write

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

The encoding process is to construct a message polynomial $m(x)$ from a given message using the same method as Eq. (*2*) to ensure that every valid polynomial code is a multiple of a polynomial generator $g(x)$.

$$C(x) = x^{n-k}m(x) + R\left\{\frac{x^{n-k}m(x)}{g(x)}\right\} \tag{3}$$

Equation (*3*) shows the straightforward construction of CRC. Taking the input message, assemble the $m(x)$. Next, multiply by x^{n-k}, and finally, divide by $g(x)$. The remainder R forms the check bits, acting as the digest for the entire message, and will be appended to the message. The decoding process is identical to the encoding step, it separates each word received into the message and the remainder portion, and then verifies

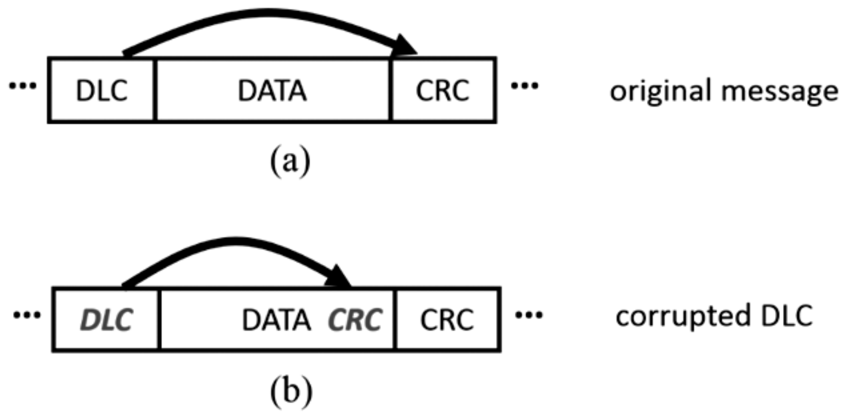

Figure 2: (a) A correct DLC with proper CRC field location.
(b) A corrupted DLC will lead to an incorrect position for CRC field.

whether the calculated remainder from the message matches the sent bits. A mismatch means that an error has occurred and the receiver will request for retransmission (ARQ) of message [8].Furthermore, CAN does not protect length field. A corrupted data length code (DLC) field will point to a wrong location for CRC or give a different result for CRC syndrome. Figures *2*(*a*) and (*b*) shows the difference of having a correct DLC field to a corrupted one.

Although this polynomial code is the basis for powerful error-correction methods and simple implementation of binary hardware, it is not suitable for protecting against intentional alteration of data. An overflow of data is also possible to occur in CRC because of its fixed number of check bits. Based on the n^{th} degree of the generator polynomial required to be attached during transmission, it reduces the transmission rate of the network. Additionally, CRC arithmetic performs XOR operation in specific values and uses shifting technique called linear feedback shift registers (LFSR) and serial transmission mode [16] that causes a slow throughput of the system. Moreover, CRC codes do not implement correction, it only enforces retransmission whenever an error is detected. Without the error correction mechanism, it is not recommended for an efficient system. Therefore, replacing this fixed 15-CRC bits by a shorter bit length of any alternative error detection method can increase the frame rate of CAN communication.

2.4. Hamming Code

This is a class of error-correcting and linear block codes used to detect and correct error bits that occur during transmission. It can also detect up to two simultaneous error bits and correct single error bit [17]. In Hamming code, the redundancy bit r ora parity bit is added to an n-bit data word (D), forming a new word of $D + r$ bits. It is described by an ordered set ($D + r$, D). The redundancy bit matrix can be expressed as:

$$[\mathrm{R}] = [\mathrm{D}] \,.\, [\mathrm{G}] \tag{4}$$

where [D] is the data matrix and [G] is the generator matrix, which is composed of an identity matrix [I] and a creation matrix [C].

$$[G] = [I : C] \tag{5}$$

The minimum number of r required for a single bit error correction is derived from the following inequality:

$$2^r \geq D + r + 1 \tag{6}$$

These redundancy bits are to be distributed at bit positions of power of 2 with the original data bits. Next, all other bit positions are assigned for the data to be encoded (for example, 3, 5, 6, 7, 9, 11, 13, 14, 15, 17, etc.). As a result, there is an increase in overhead because of interspersing the redundancy bits both for the transmitter and receiver parts.

To detect errors, the codeword vector multiplies with the transpose of the generator matrix to produce an 8-bit vector [S], known as the syndrome vector.

$$[S] = [D, R] \,.\, [G'] \tag{7}$$

If all of the elements of the syndrome vector are zeros, no error is reported. Any other non-zero result represents the bit error type and provides the location of any single bit error. It is then used to correct the original incoming data.

Hamming code is also effective on networks where the data streams are prone to single-bit errors as one of its advantages. However, if multiple errors occur, the errors can be detected but the resultant could cause another bit that is correct to be changed, causing the data to occur another error. Moreover, the redundancy bits are needed to be interspersed at power of 2 bit positions with the original data bits, which causes an overhead payload.

3. PROPOSED ALGORITHM

3.1. Enhanced-Error Detection-Correction (EEDC) codes

The proposed algorithm is a modified version of the conventional Hamming code. Each valid codeword of C bits contains the valid input data bits D_i. In any valid D_i entity, there are C bits that can be changed to give an invalid codeword. Thus, the total number of codewords corresponding to a valid data entity is $C + 1$. As there are 2^{Di} valid data patterns, the total number of codewords is $(C + 1)2^{Di}$. In D_i bit codewords, the possible number of patterns is 2^C, which limits the number of valid plus invalid codes that can exist. Thus,

$$(C + 1)2^{Di} \leq 2^C \tag{8}$$

and it can be written by

$$C = D_i + r \tag{9}$$

and

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

so that the total number of the required redundancy bits should satisfy the given condition of the inequality below

$$(D_i + r + 1) \leq 2^r \tag{11}$$

Both the data information D_i and the required r will be constructed into a polynomial form with a degree of n-1, such as

$$D(x) = \sum_{i=0}^{n-1} D_i x^i \tag{12}$$

and

$$r(x) = \sum_{i=0}^{n-1} r_i x^i \tag{13}$$

Then, the degree of polynomial $D(x)$ will increase with the nth value of r, such that

$$G(x) = D(x) \,.\, X^n \tag{14}$$

Thus, to complete the form of EEDC codes, it follows Eq. (15).

$$EEDC\ codes = G(x) + r(x) \tag{15}$$

Let us consider using this proposed EEDC codes in a 7-bit data information 1001110. Its polynomial form is $X^6 + X^3 + X^2 + X$. Therefore, the least number of r to satisfy the above inequality of (8) is 4. Its polynomial form is $r_3X^3 + r_2X^2 + r_1X + r_0$, while the polynomial in (11), $G(x)$, is $X^{10} + X^7 + X^5 + X^4$.

Next, identify the appropriate bit of the redundancy bits. Considering the above example, r_3, r_2, r_1, and r_0 are in positions 8, 9, 10 and 11, respectively. Below are the solutions in identifying the values of these redundancy bits.

The check bits in position 8 are 1, 3, 5 and 7 positions. Even parity, so r_3 is set to 0.

The check bits in position 9 are 2, 3, 6, and 7 positions. Odd parity, so r_2 is set to 1.

The check bits in position 10 are 4, 5, 6, and 7 positions. Odd parity, so r_1 is set to 1.

The check bits in position 11 occurs only on the redundancy bits r_3, r_2, and r_1. Even parity, so r_0 is set to 0.

Therefore, the EEDC codes for this 11-bit data to be transmitted as $X^{10} + X^7 + X^5 + X^4 + X^2 + X$in polynomial form is equivalent to 1001110$r_3r_2r_1r_0$, where the bit values of r_3, r_2, r_1, and r_0 are 0110respectively. The required redundancy bits are to be appended at bit positions 8, 9, 10, and 11. Thus, the 11-bit data to be transmitted will be 1001110**0110**.

4. EXPERIMENTAL RESULTS

To validate our designed algorithm and to measure the actual performance on hardware, we implemented the design in a low-power Xilinx Virtex5 FPGA by using Xilinx ISE. We chose the Spartan 6 because it is a low-cost and low-power device, which would be a likely choice for an automotive implementation.

To test the network aspects, we emulated a CAN bus in the FPGA using captured raw bus transactions from a real CAN network communicating using a predefined CAN information. These information are stored in onboard memory. The tested information is replayed to create an accurate transactions on the bus. The modified communication controller is tested into the CAN bus and configured with the same CAN parameters.

Table 2 shows the comparison results of the simulated transmission rate between CRC, Hamming codes and the proposed EEDC in an 8-bit to 8-byte data. It was tested into three implementations to verify which appropriate error detection-correction codes will give higher data rate. It shows that the transmission rate of the proposed implementation is faster as compared to CRC and Hamming codes. Moreover, the designed hardware was subjected to a random input data to analyze the results where the codes will detect more corrupted information. Figure 3 shows the error detection performance of this proposed algorithm is better to compare with CRC and Hamming codes.

Table 2
Data rate comparison between CRC, Hamming and the proposed EEDC codes.

		Data rate(bits per second)	
Transmitted data(bytes)	*CRC*	*Hamming code*	*EEDC*
1	16,640	19,600	21,040
2	14,280	16,392	20,032
3	12,656	14,280	15,648
4	11,232	12,496	12,800
5	10,200	11,104	11,376
6	9,256	10,000	11,184
7	8,472	9,088	10,752
8	7,872	8,400	8,952

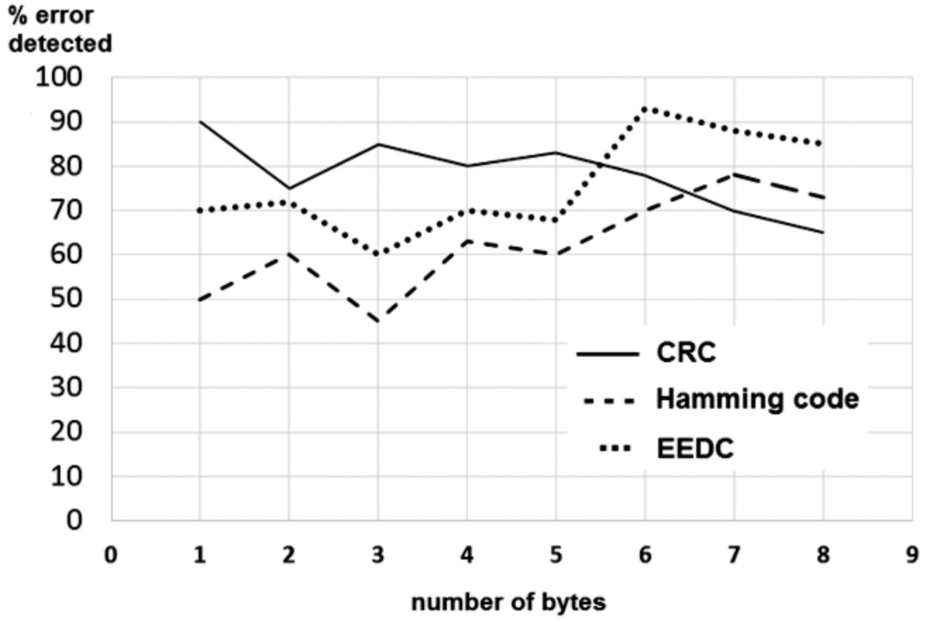

Figure 3: Error detection performance of CRC, Hamming and EEDC codes.

5. CONCLUSION

The proposed enhanced error correction and detection (EEDC) codes can be used as an alternative error detection and correction scheme for CAN protocol. The authors designed a customized CAN controller that takes advantage of the heterogeneous resources on modern FPGAs, resulting to less logic requirements and low power consumption. The experimental results showed that the objectives met its target of faster transmission rate when compared with the implementation of CRC because of the fixed data frame of the required polynomial generator and prevention of the overhead payload that exists in the conventional Hamming codes. Moreover, the study showed that the error detection performance is more efficient.

The authors plan to develop a more flexible CAN controller that are more prone to noise and errors. It should also allow the authors to investigate more advanced error detection and codes that are appropriate for CAN implementation.

ACKNOWLEDGMENT

This work was supported by the IT and R & D program of Ministry of Trade, Industry and Energy (10049192, Development of a Smart Automotive ADAS SW-SoC for a Self-Driving Car) and this this research was partially supported by the Cheongju University Research Scholarship Grants from 2015 to 2017.

REFERENCES

[1] Serfa Juan, R. O. and Kim, H. S., "Utilization of High-Speed DSP Algorithms of Cyclic Redundancy Checking (CRC-15) Encoder and Decoder for Controller Area Network", *Jurnal Teknologi*, vol.78, no. 5-9, pp 13-19, January 4, 2016

[2] Koopman, P., "32-bit Cyclic Redundancy Codes for Internet Applications", *Proc. IEEE International Conference on Dependable Systems and Networks,* 2002 http://electronics.stackexchange.com/questions/121329/whats-the-maximum-can-bus-frame-message-rate-at125-kbit-s

[3] Kim, H. S. 2016, "FPGA Implementation of Manchester Line Encoding for Frame Length Compression Scheme in CAN (Controller Area Network) Controller", *Journal of KIIT*, vol 14, no.1, pp 27-35, January 2016

[4] Abdul, R. I. and Tayel. M. B. 2015, "Simulation of Hamming Coding and Decoding for Microcontroller Radiation Hardening", *International Journal of Electrical, Computer, Energetic, Electronics and Communication*, vol. 9, no. 2, 2015

[5] KVASER, The CAN Protocol Tutorial

[6] CAN Specification Version 2.0, *The Bosch IC Design Center. Reutlingen. Germany*

[7] Held, G., 'Inter- and Intra-Vehicle Communications", *CRC Press,* 2007

[8] Voss, W., "Error Detection and Fault Confinement, in A Comprehensive Guide to Controller Area Network", *Copperhill Media Corporation,* 2008

[9] Pfeiffer, O. and Keydel, C., "Underlying Technology: CAN. In Embedded Networking with CAN and CANopen", Copperhill Technologies Corporation, 2008

[10] B. A. Forouzan., "Data Communications and Networking", 4th Edition, *Tata McGraw – Hill Education,* 2006

[11] S. Hekmat., "Communication Networks", available online: www.pragsoft.com/books/CommNetwrok.pdf

[12] Error Detection and Detection. Logic and Computer Design Fundamentals 3rd Edition, Pearson Education, 2004

[13] Kumar, U. K. and Umashankar, B.S., "Improved Hamming Code for Error Detection and Correction", *In Proc. IEEE, 2nd International Symposium on Wireless PervasiveComputing Conference*, February 5-7, 2007