International Journal of Control and Automation
Vol.10, No.4 (2017), pp.49-58
http://dx.doi.org/10.14257/ijca.2017.10.4.05

FPGA Implementation of Bit Stuffing Reduction Algorithm for CAN Communication by Miller Line Encoding Scheme

Ronnie O. Serfa Juan[1], and Hi-Seok Kim[2*]

[1, 2]Department of Electronic Engineering
Cheongju University
Cheongju, Korea
([1]ronnieserfajuan, [2]khs8391)@cju.ac.kr*

Abstract

Controller Area Network (CAN) protocol uses Non Return-to-Zero (NRZ) line encoding scheme, but it causes long runs of consecutive bits with the same signal level that leads to problems like the unsynchronized transmission. Bit stuffing is a method used by CAN to force synchronization, however, it reduces the frame rate, and causes jitter in communication. This paper aims to minimize the use of stuffed bits and to increase the CAN's frame rate. Moreover, to implement an alternative line encoding scheme called the Miller Line Encoding. It overcomes the existing drawbacks in NRZ. This paper comprises of three designed components in the proposed architecture. A counter that determines the number of zeros and ones, and produces a signal indicating when the number of 1s and 0s is odd or even. It also supplies a clock signal as a tracker to the input clock terminal. Next, a delay circuit that enters the input data streams and its output will be fed to the Miller encoder. And, a Miller encoder, designed to follow the proceeding rules: logic 1 causes a transition from one level to the other one in the middle of a bit period; logic 0 that follows logic 1 occurs no transition on the signal level; logic 0 follows another logic 0 causes a transition to the other level at the beginning of the bit period. The proposed scheme is synthesized on the Xilinx Virtex-5 FPGA. The results show that the Miller Line Encoding is better than the original line encoding.

Keywords*: Miller Line Encoding, bit stuffing, jitter, CAN, frame rate*

1. Introduction

The requirements for Controller Area Network (CAN) in real time applications needs to minimize or totally eliminate certain adversity in road accidents. Therefore, the CAN protocol in automotive industry focuses on a high level of error prevention and ensuring the reliability but maintaining the low-cost and light weight broadcast [1] set-up especially in advanced driver-assistance systems (ADAS) application that has been changing how cars are operated. It implements advanced capabilities such as automated parking, pedestrian detection with automatic braking, and other safety features. It aims to assist drivers and eliminate errors, including blind-spot and lane-departure warning systems, automatic emergency braking, lane-change assistance, *etc.* In able to cater all of the data gathered from the network, it demands an advanced performing CAN controller. This protocol is effectively used for achieving higher data rate [2]. Moreover, the CAN controller should have an efficient implementation of higher level protocols without affecting the performance of the communication controller.

The CAN protocol uses a lot of error managing schemes like the Cyclic Redundancy Checking (CRC) code as CAN's self-error detecting method for error checking on each frame's contents [3]. Also, CAN implements the bit stuffing algorithm due to insufficient

[1] ⋆Corresponding Author

signal edges for synchronization of the bit stream, bit stuffing is required [4] because it uses Non Return-to-Zero (NRZ) line encoding for data transmission and it ensures compact messages with a minimum number of transitions and high adaptability on external disturbances. In CAN standard, it allows only 5 consecutive bits of the same polarity between the Start of Frame (SOF) to Cyclic Redundancy Check (CRC) field; every bit stream of more than 5 bits of the same polarity, dominant (logic 0) or recessive (logic 1), within these frames, are considered on error condition [5]. Although bit stuffing guarantees the receiver in the synchronous state, especially when the data stream sent a large volume of data in the same polarity, either dominant or recessive manner that has no transition of the signal, it minimizes the frame rate of CAN system.

The stuffed bits in CAN communication may be as high as 19-bits and 23-bits in 2.0-A Frame and 2.0-B Frame, respectively. Equation 1 describes the worst case scenario of a number of stuffed bits for both frames.

$$stuff\ \ bits = \left\lceil \frac{34+8*(l)}{5} \right\rfloor \tag{1}$$

where l is the data length in bits and it is measured between 1 to 8 in bytes.

Because of this bit stuffing, the exact duration of the transmission of any transmission of any given frame depends not just only on the size of the payload or message frame but also on its content. Therefore, as a consequence, the reception times for messages in the same message stream may suffer from unwanted jitters [6]. Also, the timing accuracy of the networked devices in the system is affected by this variation in latency. In this paper, an alternative encoding scheme will be presented that has been analyzed to minimize frame length of CAN system and preventing the usage of any stuff bits. This paper aim to reduce the cause of the timing variation due to message length variation causes by bit-stuffing in CAN protocol.

Moreover, the purpose of this paper is to show the advantage of configurable CAN controller that can be implemented using off-the-shelf controllers. FPGA based communication controller implementation is a different approach that is ideally implemented on certain or dedicated applications but preserving the necessary protocols and it reduces power consumption because of enhanced hardware. Eliminating unnecessary blocks in communication controller minimizes the power usage and hardware utilizations. In this paper, we present an optimized CAN controller that integrates configured extension that add-ups the capabilities beyond those defined by the CAN standard.

The rest of this paper is organized as follows. Section 2, describes some existing related works and few line encoding schemes. Section 3 briefly discuss the CAN Frame Rate and its effect on the CAN system. In Section 4, describes the proposed algorithm and architecture of this paper using Miller Line Encoding. Section 5 shows the experimental results and interpretation of data. Finally, Section 6 concludes this paper.

2. Line Encoding Techniques and Related Works

Sending binary data through a network requires that the physical quantities' characteristics that should match the medium capability. Also, some factors should be considered when deciding a line encoding like self-synchronization for clock synchronization, a long string of consecutive 1s and 0s should not cause a problem in clock recovery. And, the transmission bandwidth needs to be sufficiently small compared to the channel bandwidth so that inter-symbol interference will not be a problem. Having a low probability of error is also includes because when the received signal is corrupted by noise, the receiver can easily recover the corrupted signal. Combined error detection and correction capability is an important factor because line codes required to have an error detection and an error correction capabilities. Below are the few commonly used encoding techniques

2.1. Polar Non Return-to-Zero (NRZ) Code

A polar NRZ signal is also called an NRZ-L (L for the level) signal because of a high voltage level corresponds to a positive logic level. This format is the general representation of binary data (*i.e.*, a logical zero state is transmitted as one signal level, and a logical one state as another level). These levels change at bit edges only if the bit changes, the signal waveform of NRZ is shown in Figure 1 (a).

2.2. Manchester Code

Manchester encoding is also called bi-phase encoding or phase encode. It is generated from NRZ data by a binary XOR with a clock signal which translates into a raised edge when the bit value is zero and falling edge in the opposite case. The coded data has a transition in the middle of every bit, and the direction of this transition indicates a binary zero or one. Therefore, Manchester code is known as the simplest form of binary pulse position modulation which monitors message bits to pulse position [7]. The signal waveform of this line encoding is shown in Figure 1 (b).

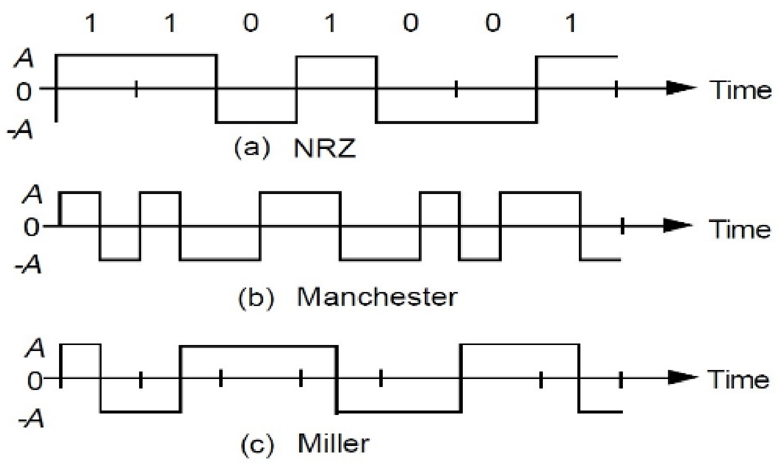


Figure 1. Signal Waveform of a) NRZ, b) Manchester and c) Miller

2.3. Miller Line Encoding

Miller coding is also called delay modulation. A transition occurs at the mid-point of each symbol interval for a “1” signal, then for a 1 followed by a 1 signal, no transition occurs at the symbol interval. No transition occurs at the mid-point of each symbol interval for a “0”. While for a 0 followed by a 0, a transition occurs at the symbol interval. Then, for a 0 followed by a 1 or a 1 followed by a 0, no transition occurs at the symbol interval [8]. Since Miller encoding is similar to Manchester encoding, except that a transition occurs in the middle of an interval only when the bit is 1, which allows higher data rates. This is shown in Figure 1 (c). Miller coding is also called delay modulation.

Reference [9] shows that the technique used is to remove certain drawbacks associated with Bipolar and Manchester coding using a different D.C. levels for representing the bit streams of NRZ, however, it only focuses analysis on the power spectral density of the encoded signal, and it did not describe the algorithm of this multilevel NRZ. Moreover, the Miller line encoding technique in not included in this paper. In [10], some line encoding formats are simulated and the designed algorithms are implemented and simulated on Xilinx design tools and the hardware abstraction is completed on Spartax-6, but, the hardware descriptions and utilizations are not presented in this paper. Paper [11] discusses various line encoding schemes that aim for an efficient transmission. Although it was simulated using Xilinx Spartan 6 but the designed architecture is nor described in this research paper. Lastly, in [12] shows a unique implementation that uses Manchester line encoding to increase the frame rate of the CAN system, again, there is no hardware utilizations describes here.

3. Controller Area Network's Frame Rate

CAN implements the bit-stuffing protocol that no more than five consecutive bits with the same polarity are transmitted on the bus. A bit is stuffed to provide the synchronization both for receiver and transmitter. This bit stuffing is necessary between frames of Start of Frame (SOF) to 15-bit Cyclic Redundancy Checking (CRC) code. In this bit-stuffing process, the number of required stuffed bit depends on the incoming bit pattern. Therefore, the output frame length is a function of the incoming bit pattern. Also, when CAN transmission starts, the CAN message cannot be interrupted, and the variation in transmission time has a potential impact on the real-time behavior of the system. The variation in message response time is referred to as timing error or "jitter". Jitter has a key impact on the performance of many applications, particularly those involving data acquisition, data processing, and control.

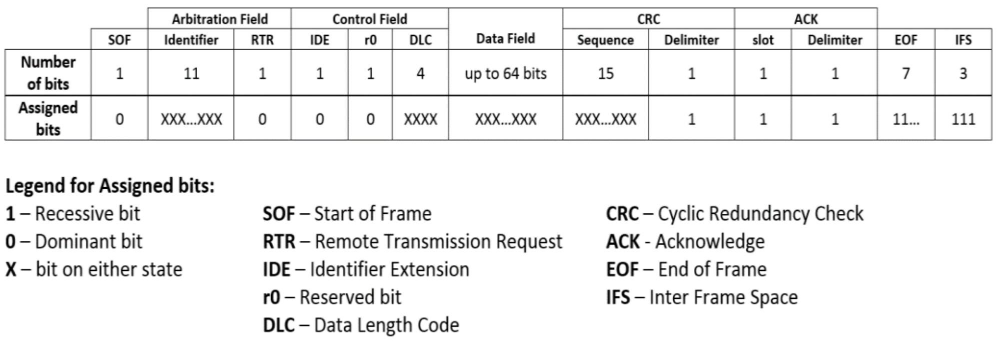
In [13], the CAN bus frame rate is defined by a given baud rate divided by the total number of frame bits and it measures in frames per second as shown in Equation (2).

$$frame\ rate\ (FR) = \left\lfloor \frac{B_R}{T_{FB}} \right\rfloor \tag{2}$$

where B_R is the baud rate of the controller and T_{FB} is the total number of frame bits.

Ideally, without the required stuff bits, the maximum frame rate of 954 frames per second for a 2.0-B Frame, it includes the inter-frame space bits at a baud rate of 125 kbps. However, in practice, it's hard to get 100% bus utilization and in NRZ line encoding scheme, definitely, stuff bits is necessary. Therefore, only 811 frames per second in a worst case scenario can be achieved. Therefore, minimizing the message frame by reducing or totally eliminates bit stuffing in CAN communication may increase the frame rate and a jitterless communication can be obtained.

Nolte [14], presented a technique to minimize the usage of the required stuffed bits, however, this masking technique may add-up to reduce the transmission rate because of additional necessary steps in able to perform the masking technique. A new approach for bit stuffing technique is presented in [15], it minimizes the usage of stuffed bits It uses a CAN analyzer to determine the suitable mechanism in reducing the stuffed bits according to frame size. But, it covers on a large data, the CAN system's frame is not always fixed to a particular frame size. Since the CAN controller stores received bits and be fetched by the host processor [16], a configurable mechanism is possible to be added in the FPGA-based communication controller in able to maximize the potential of this controller and an appropriateness for advancing the capability of this proposed paper. The Miller line encoding is selected because of the presentation presented above that have an advantage over the conventional line encoding and to other proposed papers.



		Arbitration Field		Control Field			Data Field	CRC		ACK			
	SOF	Identifier	RTR	IDE	r0	DLC		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 2. Standard CAN Data Frame (2.0-A Frame)

4. Proposed Miller Line Encoding Scheme

Miller encoding can be used for higher operating frequency and it is similar to Manchester encoding except that the transition occurs in the middle of an interval when the bits is 1. Using the Miller encoding, noise interference can be reduced.

The proposed architecture of Miller Line Encoding is shown in Figure 3. This will be added to the FPGA based communication controller as part of the reconfigured scheme to maximize the efficiency of the CAN controller. The data stream is input to a delay circuit and a counter. The counter counts the number of zeros and ones and produces a signal indicating whether the number of 1s and 0s is odd or even. The counter also supplies the clock signal to tracker circuit at a clock input terminal. The output signal from the delay circuit is applied as an input to the designed Miller encoder. The information is enclosed in the transitions of the signal level.

Also in this figure, it shows the following settings Mode = 1, CLR = 0 and CS = 1. It shows that the output of the proposed system depends on the control signal (CS). If CS is 0, the output is Miller line encoding.

The proposed implementation of this algorithm is described as follows:

The information in a Miller Line Encoding Scheme is enclosed in the transition of the signal level. The following rules are:

- A logic 1 causes a transition from one level to the other one in the middle of the bit period.
- A logic 0 following a logic 1, causes no transition.
- A logic 0 follows another logic 0 causes a transition to the other level at the beginning of the bit period.

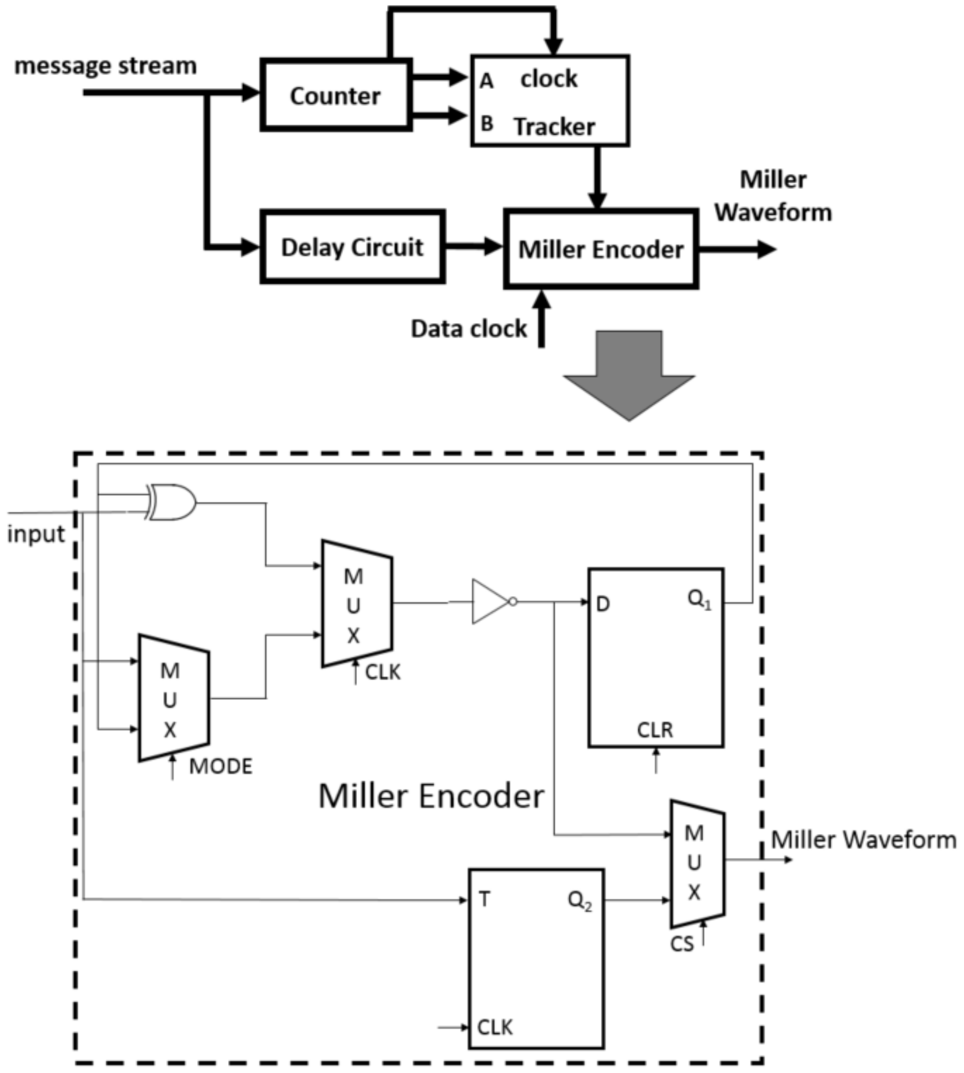


Figure 3. Proposed Architecture of Miller Line Encoding

The four states are shown in Figure 4 as 00, 01, 10, and 11. The transition is based on logic 1 and logic 0. Below is the discussion of this finite state operation.

In the initial state, reset is 1.

Then, the next state will be 00, and after this reset, it is set always on zero level.

When the input is 0, and the current state is 00, the next state is 01.

When the input is 1, and the current state is 00, the next state is 10.

If the input is 0, and the current state is 01, the next state is 10.

And, if the input state is 1, and the current state us 01, then, the next state is 01.

And, when the input is 0. And the current state is 10, the next state will be 11.

If the input is 1, and the current state is 10, the next state will be 10.

If the input is 0, and the current state is 11, then, the next state will be 00.

And, if the input is 1, and the current state us 11, the next state is 01.

Table 1 shows the state status of this Finite State machine for Miller Encoder.

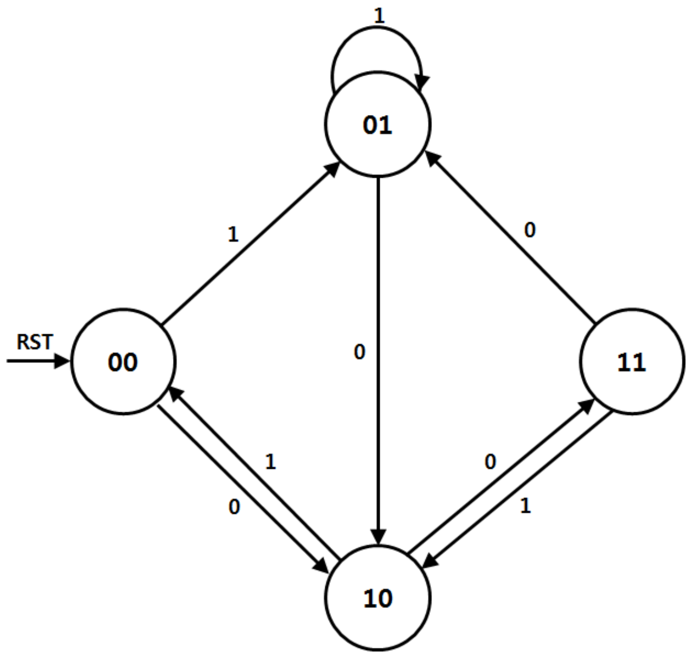


Figure 4. Finite State Machine for Miller Encoder

Table 1. Status Condition of Miller Encoder

Reset	Input	Current State	Next Sate
1	-	-	00
0	0	00	01
0	1	00	10
0	0	01	01
0	1	01	11
0	0	10	11
0	1	10	10
0	0	11	00
0	1	11	01

5. Experimental Results

The proposed implementation was synthesized to Xilinx Virtex 5 FPGA by using Xilinx ISE. Table 2 shows the result of device utilizations from the Xilinx synthesis tool. It shows that the utilization of resources in the number of slice LUT and number of paired flip-flops decreases when this implementation is being utilized to 17% and 29 %, respectively. Figure 5 shows the simulation results in using Verilog HDL, the 44 input bits was simulated and its shows that in “data_input” and the clock is given as a rising edge which produces the output as miller_out.

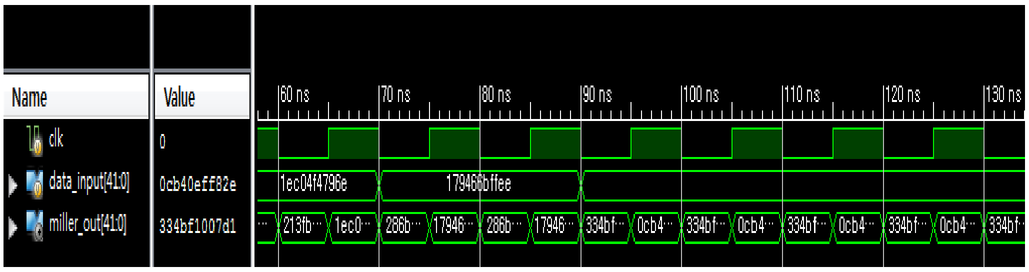


Figure 5. Simulation Result of Miller Line Encoding

Table 2. FPGA Device Utilization Summary

Resources		**Used**	**Available**	**Utilization**
Number of Slice LUTs	NRZ	331	12,480	2%
	Miller	275	9,854	2%
Number of fully used LUT-FF pairs	NRZ	212	22,432	1%
	Miller	150	20,143	1%
Number of bonded IOBs	NRZ	14	341	2%
	Miller	15	312	3%

If Miller line encoding is implemented, stuffed bits can be eliminated, Table 3 shows the comparison of Frame length variations between NRZ and Miller line encoding, both for frames 2.0-A and 2.0-B. The total number of frame bits decreases by 14% and 15%, respectively on an 8-bytes data bit stream.

Table 3. Number of Stuffed Bits Utilized between NRZ and Miller Line Encoding

	NRZ		**Miller**	
	2.0-A Frame	**2.0-B Frame**	**2.0-A Frame**	**2.0-B Frame**
	Data Field 8 bytes	Data Field 8 bytes	Data Field 8 bytes	Data Field 8 bytes
Number of Frame bits (without stuffed bits)	111	131	111	131
Maximum bit stuffing (worst case)	19	23	0	0
Total number of frame bits	130	154	111	131

Table 4 shows the Miller Line Encoding minimizes the frame payload and resulting to an increase in the frame rate of the controller, both frames 2.0-A and 2.0-B uses a 125 kbps of baud rate at 8-bytes data bit stream.

Table 4. Frame Rate Comparison Using NRZ and Miller Line Encoding both for Frames 2.0A and 2.0B

	NRZ	Miller
Frame	**Frame rate (frame per second)**	**Frame rate (frame per second)**
2.0A	984	1,224
2.0B	811	1,023

6. Conclusion

In this paper, we have given an overview of the required system to achieve a perfect error-free transmission. However, it cannot be attained in real time application especially if the transition density on the data sequence during the long run of 0s or 1s. Miller Line Encoding benefits to eliminate the drawbacks of the aforementioned scheme such as the bit stuffing process that causes jitter in CAN communication. Also, by modifying the FPGA-based communication controller can enhance the system into more advanced capabilities to target in reducing the CAN's frame length and aiming to increase the frame rate of CAN controller. In Tables 2 and Table 3 present that the performance of Miller Line Encoding, which is much better than NRZ line encoding. Moreover, the system's stuffed bits and overhead payload decreases.

In future, we aim to investigate extending this communication controller in more flexible applications. We intend to develop a reconfigurable hardware that are energy efficient and that will allow us to explore more advanced network set-ups.

References

[1] H. Chen, and J. Tian, "Research on the Controller Area Network", International Conference on Networking and Digital Society, vol. 2, (**2009**), pp. 252-254.

[2] B. S. Rao, "Controller Area Network for Monitoring and Controlling the Environment Parameters using Zigbee Communication", International Journal of Advanced Engineering Technology, vol. 3, issue 2, **(2012)**, pp. 34-36.

[3] 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, vol. 5-9, no. 78, **(2016)**, pp. 13-19.

[4] W. Voss, "Controller Area Network, Serial Network Technology for Embedded Solutions", ESD Electronics, **(2008)**.

[5] W. Voss, "A comprehensive Guide to Controller Area Network" ESD Electronics, **(2008)**.

[6] Numgi Kim, "Variable CRC Scheme for Efficient Data Transmission Control for IEEE 802.16e Wireless Network", JKIIT, vol. 7, no. 3, **(2009)**, pp. 109-115.

[7] H. Bidgoli, "Handbook of Computer Networks: Key Concepts, Data Transmission, and Digital and Optical Networks", **(2008)**.

[8] W. C. Lindsey and M. K. Simon, "Telecommunication Systems Engineering", Prentice-Hall, **(1973)**.

[9] V. Kulkarni, P. N. Arya and P. V. Gaikar, "A Multilevel NRZ Line Coding Technique", International Conference on Technology Systems and Management (ICTSM) 2011 Proceeding, International Journal of Computer Applications, **(2011)**, pp. 17-22.

[10] A. K. Singh, K. P. Pandey, V. Singh, "Representation and Conversion of Line Coding using Mealy Sequential Network", International Journal of Emerging Trends in Engineering and Development, vol. 1, no. 6, **(2016)**, pp. 76-81.

[11] V. Singh, and B. Mishra, "FPGA Implementation of Various Lines Coding Techniques for Efficient Transmission of Digital Data in Communication", International Journal of Research in Engineering and Technology, vol. 3, no. 4, **(2014)**, pp. 60-63.

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

[13] http://electronics.stackexchange.com/questions/121329/whats-the-maximum-can-bus-frame-message-rate-at125-kbit-s.

[14] T. Nolte, "Using bit-stuffing distributions in CAN analysis", IEEE Real-Time Embedded Systems Workshop, **(2001)**, pp. 1-6.

[15] M. M. Hassan, "Bit stuffing techniques Analysis and a Novel bit stuffing algorithm for Controller Area Network (CAN)", International Journal of Computer System, vol. 2, no. 3, **(2015)**, pp. 80-87.
[16] A. S. Shinde, V. B. Dharmadhikari, "Controller Area Network for Vehicle Automation", International Journal of Emerging Technology and Advanced Engineering, vol. 2, no. 2, **(2012)**, pp. 12-17.

Authors

Ronnie O. Serfa Juan (M'15), received his BSc in Electronics and Communications Engineering from the Technological University of the Philippines-Manila, and he earned his MSc in Information and Telecommunications Studies, majoring in Computer Systems and Network Engineering, at Waseda University in Tokyo, Japan in 1999 and 2007, respectively. He is currently working toward his Ph.D., majoring in Computer and Control, at Cheong-Ju University in Cheong-Ju City, South Korea. He passed the ASEAN Electronics Engineering evaluation examination last November 2016. His research interests include radio frequency identification (RFID), advanced driver assistance system (ADAS) technology, Controller Area Network (CAN) and FlexRay Technology.

Hi-Seok Kim (M'06), received his Bachelor of Science, Master of Science and Ph.D. in Electronic Engineering from Hanyang University, the Republic of Korean in 1980, 1985 and 1987 respectively. He is currently a Professor in Electronic Engineering Department, Cheong-Ju University, Cheong-Ju City, South Korea. His research of interests includes digital video/audio system design, multi-view imaging, 3D image processing, and FPGA design. Dr. Kim has served as a General Chair and a committee member of many Korean and International Conferences, including International SoC Design Conference and IEEE ISCAS 2012.