International Journal of Computer System (ISSN: 2394-1065), Volume 02– Issue 03, March, 2015
Available at http://www.ijcsonline.com/

Bit stuffing techniques Analysis and a Novel bit stuffing algorithm for Controller Area Network (CAN)

Maha Medhat Hassan

System and Computers Department, Al-Azhar University, Cairo Egypt

Abstract

A new approach for Bit stuffing techniques Analysis is introduced. CAN bit stuffing analyzer is designed and implemented and the different techniques for minimizing stuffing-bit are evaluated using 50,000 random data field frames, illustrating how the payload size affect the number of stuffed bits. The expected number of stuffed bit per data field has been also determined. The new CAN analyzer helps to determine the suitable mechanism for minimizing stuffing-bit in CAN messages according to frame size.XOR mechanism has a good effect on large size of data, since it provides about 25 % reduction in stuffing bits, and the maximum number of stuffed bits in individual frame reduced significantly in the data field size from(4-8) bytes. A novel Inversion Bit Stuffing Mechanism (IBSM) has been introduced to reduce the frequently stuffed bits, where the data bytes (2-5) have only one stuffing bit (in average) which tends to decrease the transmission time. Finally careful priority usage was issued to avoid stuffed bits in the identifier field.

Keywords: Bit *stuffing analysis, CAN analyzer, Careful priority usage, Inversion bit stuffing mechanism*.

I. INTRODUCTION

Controller Area Network (CAN) [1] is a serial communication protocol technology that was originally designed for the automotive industry, especially for European cars, but has also become a popular bus in industrial automation as well as other application [1] – [2]. CAN use „„Non Return to Zero" (NRZ) [3] coding for bit representation, In NRZ coding there is no easy way to tell where each bit starts or ends when there is a long sequence of "0" or "1" has been transmitted. If this type of situation arises it may happen that transmitter and receiver lose synchronization. Such a drift might, in turn, result in a message corruption. To avoid the possibility of this scenario, the CAN communication protocol at the physical level employs a bit-stuffing mechanism [1], which operates as follows: after a five consecutive identical bits have been transmitted in a given frame, the sending node adds an additional bit, of the opposite polarity Fig.1.

All receiving nodes must remove the „stuffed" bits to recover the original data, bit stuffing is also employed to remove bit-patterns that are used by the CAN protocol for error signaling: such patterns normally contain sequences of consecutive identical bits [3].

This stuffed bits increase the CAN frame length and accordingly increase the transmission time, since the total number of bits in a CAN frame before bit-stuffing is

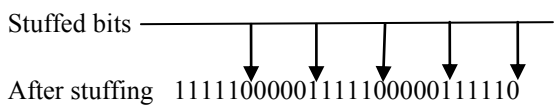

Figure 1: The basic operation of bit-stuffing

(47+8s) where (s) is the number of bytes of payload data (s ϵ [0, 8]), the frame layout is defined such that only 34 of these 47 bits are subject to bit-stuffing [4].

The total number of stuffed bits can be calculated by the *equation (1)* [9]:

$$No.of\ stuffed\ bits = \left\lfloor \frac{\text{No. of bits subject to bit stuffing} - 1}{Max.No.of\ consecutive\ bits\ bits\ allowed - 1} \right\rfloor \quad (1)$$

The total number of frame length after bit-stuffing can be calculated by *equation (2)*

$$8S + 47 + \left\lfloor \frac{34 + 8S - 1}{4} \right\rfloor \quad (2)$$

That means if the frame payload has 8 bytes data, and at the worst case bit stuffing, the number of bit stuffing is equal to 24 bit, which represent about 22% of the total size of CAN frame standard format, as an example if the CAN operating at 125 Kbit/s baud rate, the stuffed bit require 0.192 ms plus the necessary time of .888ms to transmit the original frame.

When attempting to reduce message length and reduce the number of stuffed bits techniques described by Nolte [4,5], in that scheme, the CAN frame is XOR-ed with the bit-pattern (101010….) Fig.2, the receiver must apply the same bit operation to extract the original data.

Original frame:	000000111110011000000111
XOR with bit mask:	101010101010101010101010
Transmitted frame:	101010010100110010101101

Figure 2: XOR-ing the frame with special bit pattern

Nolte has examined 2500 CAN frames from automotive system, and found that the probability of having bit value of „„1" (or„„0") in the data segment is not 50%. More specifically, he observed that the probability of having consecutive bits of the same polarity was high, and - therefore – the number of stuff bits is higher than that would be expected with random data. To remove these sequences of ones and zeros that can be highly found in a real CAN traffic, Nolte presented this simple XOR coding.

Nolte approach was subsequently extended with Nahas's pilot study in selective XOR[6], where the frames were tested frame-by-frame and only where (hardware)bit-stuffing would be applied – to the „„raw" frame – the frame is subject to XOR transform.

To prevent hardware bit stuffing, higher performance was achieved by means of software bit stuffing(SBS)[7]. In this technique data frames are checked before transmitting. If a sequence of four consecutive bits is detected, SBS adds an additional bit of opposite polarity. Here at most five consecutive bit of same polarity is allowed (one stuff bit and four in coming data bit), but this technique increases the total frame length and the transmission time, where the 5 byte data require 13 stuffed bits in the worst case compared with only 9 stuffed bits at maximum without SBS.

In another approach K. Park and M. Kang[8] have issued that, Nolte XOR-ed method to reduce frame sizes, and reduce the length of CAN messages may change the message priority because the problem of priority inversion has not been addressed by Nolte.

Example of priority inversion is shown in Table1.

Table .1. Example of priority inversion problem in the identifier field.

	High priority CAN ID	Low priority CAN ID	Description
Original CAN message	00001010111	01111110101	Low bit value has higher priority
XORing (by Nolte et al.)	010<u>11111</u>01	00101011111	Mask:0101010101
Bit stuffing message	01011111<u>0</u>101	001010111110	Priority inversion occurs

The rest of this paper is organized as follows: in section II, a new a CAN analyser has been designed, implemented and verified. The application of the new analyser on the data field of the CAN frame for different bit stuffing techniques and comparative study are given in section III. In section IV a novel Inversion Bit Stuffing Mechanism (IBSM) has been introduced to reduce the frequently stuffed bits. Reducing stuffed bits in the identifier field is addressed in section V, and last, section VI draws the conclusions.

II. BIT STUFFING CAN ANALYZER

A. CAN analyzer usage and flow chart

In the last few years many techniques have been proposed to reduce length variations to is low level in the CAN message transmission time caused by bit stuffing[10], starting with nolte's technique[4-5] which has suggested XOR-ing the CAN frame with the bit pattern 101010……, To remove sequences of ones and zeros that can be highly found in a real CAN traffic, in later pilot study[6] has issued that nolte's technique has a good effect only in certain sequence of data, For example, if a general CAN message is modeled using random data, then use of the Nolte (XOR) transform would not be expected to produce a significant reduction in the level of bit-stuffing, then a selective XOR has been suggested to improve nolte's technique, but the problem in selective XOR is one bit is required in order to specify whether or not the selective XOR is applied to the payload; in practice, this means that one byte is wasted[11].

In this paper we would like to examine the effect of this different techniques on the number of stuffed bits reduction for different size of data (1-8) bytes, we implement a CAN analyzer using C language which generate random data and then check for bit stuffing to determine the number of stuffed bits per data frame showing the effect of the previous different techniques, to determine the suitable mechanism for minimizing stuffing-bit in CAN messages according to frame size(1-8) data bytes.

The CAN analyzer is running 8 times to examine the stuffed bits for different size of data (1-8) bytes that may be included in the CAN payload, in each time the analyzer used to generate and test 50,000 random data frame.

After 50,000 frame has been generated and tested , the analyzer shows the percentage of frames that required bit stuffing and the Max No. of stuffed bits in individual frame, and the Frequently No. of stuffing bits in individual frame, the analyzer is used to evaluate the different techniques, and comparison is done between all of them in the next sections.

To analyze the effect of bit stuffing on the CAN data field using random data, a new analyzer has been designed and implemented using the C language Fig.3, the program is designed to analyze the whole range of the data field, which varies between (1-8) byte, the program works as follows:

1. Determine the number of the data bytes included in the payload.
2. Run the loop for 50,000 random data field frames.
3. Generate random binary number.
4. Check for bit stuffing in the data field and calculate the number of necessary stuffed bits.
5. Finally the program out the percentage of data fields that require bit stuffing and the number of the stuffed bits.

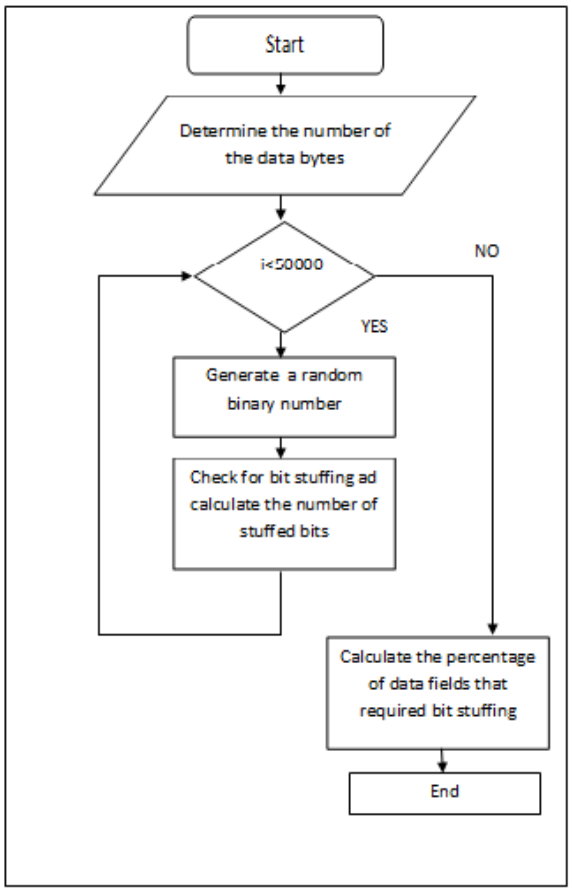

Figure 3: The Flowchart of the CAN analyzer.

B. *CAN analyzer verification*

In order to verify the new CAN analyzer results. The data given in Nolte XOR approach [5], was subject to the new CAN analyzer test, and the results are compared with Nolte results. A good agreement has been achieved as shown in Table 2.

Table 2: Expected stuffed bits with corresponding probability of occurrence for Nolte approach and new CAN analyzer.

	Probability of stuffed bits for zero byte data +CRC		Probability of stuffed bits for one byte data +CRC	
No of data bytes	0		1	
No of bits	0		8	
Total CRC+data	15		23	
No of stuffed bits	probability		probability	
	Nolte [5]	**CAN analyzer**	Nolte [5]	**CAN analyzer**
0 1 2 3 4	.676 .229 .0323 .00061 0	.608 .332 .0588 .0012 0	.485 .388 .112 .0141 .000693	.4648 .4004 .1232 .0112 .0004

III. ANALYSIS OF BIT STUFFING TECHNIQUES

In the following section the different bit stuffing techniques are subject to analysis for comparative study using the new CAN analyzer. To avoid the priority inversion problem the analysis will applied to the data field only.

A. *Traditional Bit stuffing techniques*

Data field contains from zero to eight bytes of data, and to investigate the effect of data size on the number of stuffed bits, a50, 000 random data field frames have been generated and analyzed using the proposed CAN analyzer introduced in section II.

Table 3 shows the percentage of the frames that required bit stuffing to the total number of frames, total number of bit stuffing, maximum and frequently number of stuffed bits in individual frame, compared with the maximum bit stuffing calculated by the *equation (1)* above.

Clear result are shown in figure 4, illustrating the percentage of data fields required bit stuffing as a function of the number of data byte in the CAN frame according to Table 3.

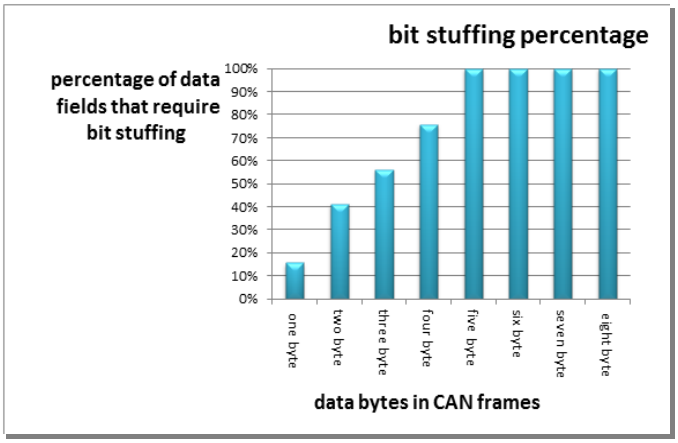

Fig.4: the percentage of data field in frames that require bit stuffing from the total CAN frames as a function of data size.

TABLE 3: Results of Bit stuffing analysis of the data field from 1 to 8 bytes in the CAN frame.

No. of data bytes	Percentage of frames require bit stuffing	Worst case bit stuffing Calculated by eq. (1)	Max No. of stuffed bits in individual frame	Frequently No. of stuffing bits in individual frame
1	15.72%	1	1	1
2	41.07%	3	3	1
3	56.06%	5	5	1-2
4	75.86%	7	6	1-3
5	100%	9	8	1-4
6	100%	11	9	3-6
7	100%	13	12	5-8
8	100%	15	13	6-9

It is clear from Table 3 and Fig.4 after examining 50,000 random data field, that:

1- As the size of data increases in the frame, the number of stuffed bits increases.
2-For CAN frames that include from 5 to 8 bytes, all the50,000 data field frames have exposed to bit stuffing, which mean thatit is preferable to send small size of data per frame(less than 5-Byte) to provide low bit stuffing.
3- The frequently stuffed bits are less than that calculated by the eq.(1)(ex. For 7- byte data field the most number of stuffed bits (5-8 bits), which are less than the maximum calculated number (13 bits)).

B. Nolte XOR Approach

To reduce the number of stuffed bits in the data field, Nolte XOR approach istested on the 50,000 random data field using the new proposedCAN analyzer introduced in section2, the analyzer was first modified to perform XOR operation to all the data fields before checking if the frame requires bit stuffing or not.Table 4 shows the impact of frame data size on thepercentage of frames that require bit stuffing, total number of bit stuffing, and the maximum and most number of stuffed bits in individual frame.The effect of Nolte - XOR operation on the reduction level of stuffing bits is shown in Fig.5.

TABLE 4: Nolte XOR approach bit stuffing analysis of thedata field from 1 to 8 bytes in the CAN frame.

No. of data bytes	Percentage of frames require bit stuffing	Worst case bit stuffing Calculated by eq. (1)	Max No. of stuffed bits in individual frame-	Frequently No. of stuffing bits in individual frame
1	15.6%	1	1	1
2	35.96%	3	3	1
3	52.33%	5	5	1-2
4	62.37%	7	5	1-2
5	75.9%	9	6	1-3
6	75.94%	11	6	1-3
7	71.54%	13	6	1-3
8	75.83%	15	6	1-3

Figure 5 shows the percentage of data fields required bit stuffing as a function of the number of data byte in the CAN frame using Nolte XORapproachaccording to Table4.

A significant reduction in stuffed bits is clear in the data size (5-8) byte after expose the data field to Nolte XOR operation.Significant reductions in the percentage of frames that required bit stuffing and in the maximum number of stuffed bits are achieved by comparing Table 3 and Table 4, as shownin Table 5, and Fig.6.

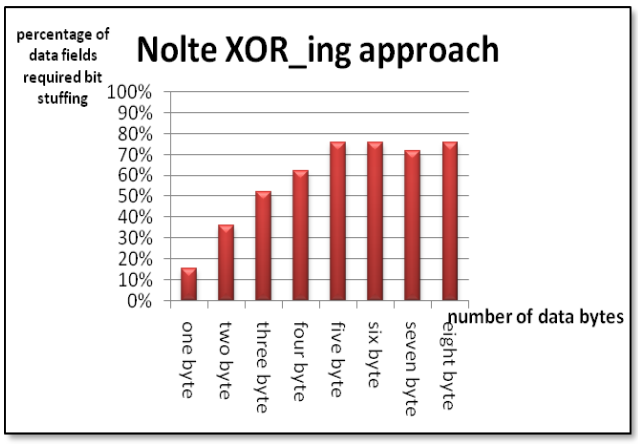

Figure 5. the effect of XOR operation on the reduction level of stuffing bits.

TABLE 5: Impact of Nolte XORoperation on percentage and Numbers of stuffed bits in CAN frames.

No. Of data bytes	Reduction in the Percentage of frames require bit stuffing	Reduction in the Max number of stuffing bits in individual frame	Reduction in the frequently number of stuffing bits in individual frame
1	0.12%	zero	Zero
2	5.1%	zero	zero
3	3.73%	zero	zero
4	13.49%	1	zero
5	24.1%	2	1
6	24.06%	3	2-3
7	28.46%	6	4
8	24.17%	7	5-6

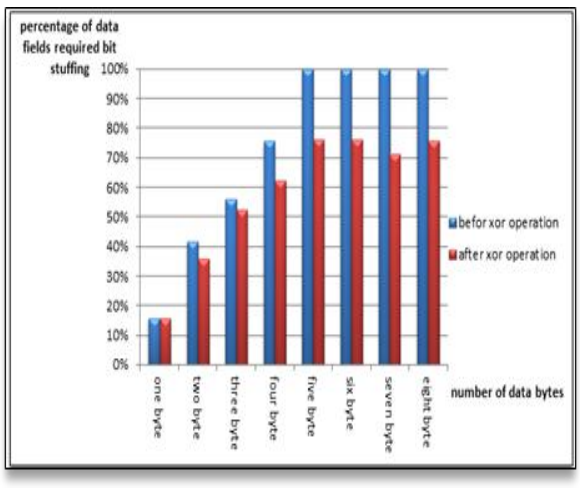

Figure 6.Graphical representation of the Impact of Nolte"sXOR operation on percentage and Numbers of stuffed bits in CAN Frames.

It is clear from Table 5 and Fig.6 after examining 50,000 random data field using Nolte XOR Approach, that:

1- Small reduction in bit stuffing in CAN frames for small data size(1-3) bytes, and no reduction in number of stuffed bits in individual frame (one byte).

2- XOR operation has a good effect on large data size (about 25 %), also the maximum number of stuffed bits in individual frame reduced significantly in the data field size from(4-8) bytes (ex. For 8 bytes data, the maximum stuffed bits reduced from 13- bits to just 6- bits).

3- Frequently numbers of stuffing bits in individual frame after XORing operation are about (1-3) bits, so the level of pessimism has been reduced, compared to the worst case bit stuffing.

C. Selective - XOR Approach

Since the reduction in bit stuffing in CAN frames for small data size is small in Nolte XORoperation [5], selective XOR[6] was introducedfor the data sizefrom 1 to 4bytes.In this technique, frames are tested frame-by-frame, and only frames that require bit stuffing is subjected to XOR operation. Random 50,000 data field frames are examinedusing the new proposedCAN analyzer introduced in section2.The analyzer was modified to test each data field first, if it require bit stuffing, the whole data field was exposed to XOR operation, otherwise the frame is left as it is without any changes, but since all the frames that include from 5 to 8 bytes have been exposed to bit stuffing as shown in table 4, so there is no meaning to apply selective XOR on the data size from 5 to 8 bytes,and only the data size from 1 to 4 bytesare analyzed, the resultshows the effect of selective XORin Table 6 and Fig.7.

TABLE 6: effect of selective XOR and reduction of bit stuffing on the data field from 1 to 4 byte

No. of bytes	Percentage of data fields require bit stuffing	Worst case bit stuffing Calculated by eq. (1)	Max number of stuffed bits in individual frame	most number of stuffing bits in individual frame
1	zero	1	Zero	Zero
2	10.58%	3	2	1
3	25.45%	5	4	1
4	44.98%	7	5	1-2

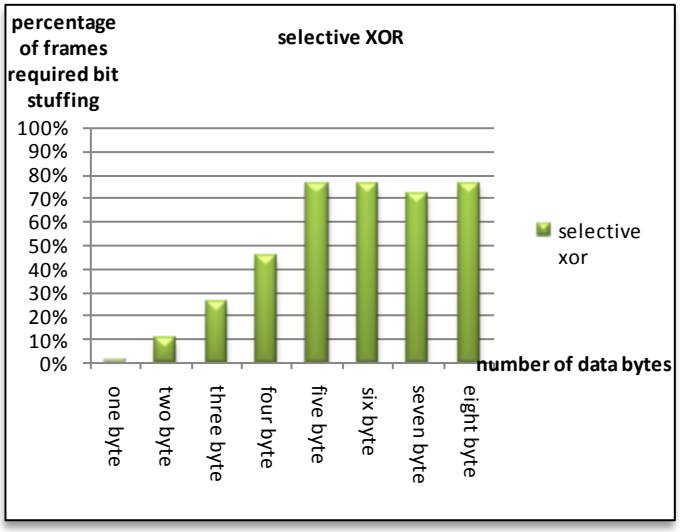

Figure 7.The percentage of data fields that required bit stuffing After selective XOR.

Selective XOR has a good effect on bit stuffing reduction, .clear result is shown in figure 8.

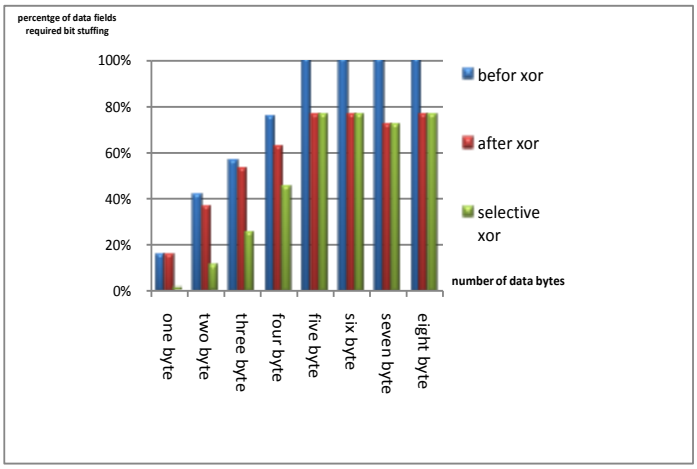

Figure 8. Performance Comparison of Mechanisms of XOR and selective XOR.

D. Inversion Bit Stuffing Mechanism (IBSM)

A novel Inversion Bit Stuffing Mechanism (IBSM) is introduced to reduce the number of stuffed bits in the data field.In this technique the data field of the CAN frame was checked byte by byte, if five consecutive identical bits have been found in any byte(11111 or 00000), the fifth identical bit is inverted(11110 or 00001),at the receiving node this bit is inverted again to avoid changing of the data, so a bit is required as a flag byte to indicate the inversion process,i.e. one byte is reserved as a flag, and seven byte is the maximum size of data sent in the CAN frame instead of .eight

Example :If the first data byte is (01111100) ,after applying (IBSM), the result will be (01111000) and the flag byte will be (10000000),indicating that the first byte in the data has been changed, applying this technique again to recover the original data, If the flag bits is set, and four consecutive identical bits is found the next bit must be inverted to be (01111100).

To evaluate the performance of this new technique, the50000 random data field has been examined for the different available data size from 1 to only 7 bytes(a byte is reserved as a flag byte), usingthe CAN analyzer introduced in section II after a modification to perform inversion process first and then check for bit stuffing,the results are shown in Table 7.The Performance comparison of mechanisms of XOR, selective XOR and the new Inversion Bit Stuffing Mechanism (IBSM)has been illustrated in Fig 9.

Table 7. The Impact of Inversion Bit Stuffing Mechanism (IBSM)on the frequently number on stuffed bits

No. of data bytes	Percentage of frames required bit stuffing	Maximum number of stuffed bits in individual frame	most number of stuffing bits in individual frame
1	zero	zero	zero
2	14.48%	2	1
3	24.96%	4	1
4	38.64%	5	1
5	68.56%	5	1
6	100%	6	1-2
7	100%	8	1-3

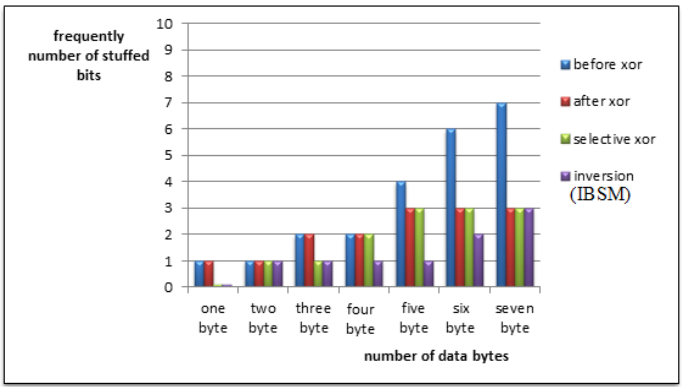

Figure 9: Performance Comparison of Mechanisms of XOR, selective XOR and the Inversion Bit Stuffing Mechanism (IBSM)

The Inversion Bit Stuffing Mechanism (IBSM) has a good effect on reducing the number of frequently stuffed bits, where the data bytes (2-5) have only one stuffing bit (in average) compared to 2 stuffing bits for four bytes data and 3 stuffing bits for five bytes datausing Nolte XORing

[5] and Selective XORing[6] techniques as observed in Fig.11.

IV. BIT STUFFING ANALYSIS IN THE IDENTIFIER FIELD

Because of priority inversion problem[8], the identifier field cannot be XOR-ed by Nolte approach[4], as an alternative technique to reduce the bit stuffing in the identifier field, Nolte has suggested a careful priority technique[5], in this paper the identifier field for both standard and extended CAN format has been analyzed, showing the percentage of frames exposed to bit stuffing and number of stuffed bits, showing that the stuffed bits in the identifier can be avoided by selecting the priorities that do not require hard ware bit stuffing.

A. Standard CAN frame has 11 bit identifier, the maximum number of stuffing bits is two, by examining the 2^{11}range of the standard frame (from 0 to 2048), the percentage of frames required (0-2) stuffing bits as shown in Table 8.

TABLE 8: Bit stuffing analysis for identifier field in Standard CAN format

No. of stuffed bits	No of furames	Percentage of frames subject to bit stuffing %
Zero	1	75.48828125%
One	4	23.2421857%
Two	2	1.26953125%

Almost 75.49 % of the identifier frames will not be subject to the bit stuffing operation. Frames with one bit stuffing rather than frames with two bit stuffing can used.

B. Extended CAN frame has 29 bit, and the maximum number of stuffed bits is 7 bits. Since the range of extended CAN is very large (from 0 to 2^{29}),A random sample of one million identifier frame has been examined, the percentage of frames required stuffed bits (0-7 bits) are shown in Table 9.

TABLE 9: Bit stuffing analysis for identifier field in extended CAN format

No of stuffing bits	**No of frames**	**Percentage %**
0	268957	26.8957%
1	409871	40.9871%
2	241123	24.1123%
3	69032	6.9032%
4	10194	1.0194%
5	811	0.0811%
6	12	0.0012%
7	0	0%

A percentage of 92% has a maximum of 2 bits stuffing, which means that the bit stuffing is rarely happen and can be avoided by selecting the appropriate identifiers.

V. CONCLUSION

A new bit stuffing analyzer is designed and implemented. The different techniques for minimizing stuffing-bits have been investigated and analyzed using the new bit stuffing analyzer on 50,000 random data field frames for both data and identifier fields separately, illustrating how the payload size affect the number of stuffed bits. The expected number of stuffed bit per data field has been also determined.

XOR mechanism has a good effect on large size of data, since it provides about 25 % reduction in stuffing bits, and the maximum number of stuffed bits in individual frame reduced significantly in the data field size from (4-8) bytes.

A novel Inversion Bit Stuffing Mechanism (IBSM) has been introduced to reduce the frequently stuffed bits .IBSM reduces the number of frequently stuffed bits, where the data bytes (2-5) have only one stuffing bit (in average) compared to 2 stuffing bits for four bytes data and 3 stuffing bits for five bytes data using Nolte XORing [5] or Selective XORing [6] techniques.

The bit stuffing for the identifier field has been analyzed for both standard and extended frame format, the whole range (0-2048) for the standard format has been examined, showing that 75.49 % of the identifier frames will not subject to the bit stuffing operation, and a sample of one million identifiers for the extended frame format was analyzed, showing thata percentage of 92% has a maximum of 2 bits stuffing, which means that the bit stuffing is rarely happen and can be avoided by selecting the appropriate identifiers.

REFERENCES

[1] Robert Bosch GmbH, CAN Specification Version 2.0, 1991.
[2] Road Vehicles - Interchange of Digital Information - Controller Area Network (CAN) for High-Speed communication. International StandardsOrganization (ISO). ISO Standard-I 1898, Nov 1993.
[3] M. Farsi, M. Barbosa, CANopen Implementation, Applications to IndustrialNetworks, Research Studies Press Ltd., England, 2000.
[4] T. Nolte, H. Hansson, C. Norstr. m, S. Punnekkat, Using bit-stuffing distributions in CAN analysis, in: IEEE/IEE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium) London, 2001.
[5] T. Nolte, H.A. Hansson, C. Norstr. m, Minimizing CAN response-time jitter by message manipulation, in: Proceedings of The Eighth IEEE Real-Time andEmbedded Technology and Applications Symposium (RTAS 2002), San Jose, California, 2002.
[6] M. Nahas, M.J. Pont, Using XOR operations to reduce variations in thetransmission time of CAN messages: a pilot study, in: A. Koelmans, A. Bystrov,M.J. Pont, R. Ong, A. Brown (Eds.), Proceedings of the Second UK EmbeddedForum 2005, Birmingham, UK, October 2005, pp. 4–17, Published byUniversity of Newcastle upon Tyne.
[7] M. Nahas, M. Short, and M. J. Pont. The impact of bit stuffing on the real-time performance of a distributed control system. In Proc. 10th International CAN conference, pages 10.1–10.7, 2005.
[8] K. Park, M. Kang, and D. Shin, "Mechanism forMinimizing Stuffing-bit in CAN Messages," The 33rdAnnual Conference of the IEEE Industrial ElectronicsSociety (IECON'07), pp. 735-737, 2007
[9]Jena, T.R. Swain, A.K. Mahapatra, K., "A Novel Bit stuffing technique for Controller Area Network (CAN) Protocol", Proceedings of the 2014 International Conference on Advances in Energy Conversion

Technologies - Intelligent Energy Management: Technologies and Challenges, ICAECT 2014
[10] M. Nahas, M. J. Pont, and M. Short. Reducing messagelength variations in resource-constrained embedded systems implemented using the CAN protocol. Journal of Systems Architecture, 55(5–6):344–354, 2009
[11] Gianluca Cena, *Senior Member, IEEE*, Ivan Cibrario Bertolotti, *Member, IEEE*, Tingting Hu, *Member, IEEE*, and Adriano Valenzano, *Senior Member, IEEE* " Fixed-Length Payload Encoding for Low-JitterController Area Network Communication", IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 9, NO. 4, NOVEMBER 2013