FPGA Implementation of Miller Line Encoding to Prevent Stuff Bits in CAN for Jitterless Communication

Ronnie O. Serfa Juan[1], Minwoo Jeong[2] and Hi Seok Kim[3]
Department of Electronic Engineering, Cheongju University
Cheongju-si, Chungcheongbuk-do 28-503, South Korea
E-mail: [1]ronnieserfajuan@cju.ac.kr, [2]jeongminwoo7@naver.com, and [3]khs8391@cju.ac.kr

Abstract. Controller Area Network protocol utilizes Non Return-to-Zero (NRZ) line encoding, but long runs of consecutive bits with the same signal level may cause problems in transmission. Adding stuff bits can force the synchronization, however, it reduces the frame rate and causes jitter in communication. The main objective of this paper is to minimize or totally prevent the usage of stuff bits that causes jitter in CAN communication. Miller Line Encoding minimizes these drawbacks. 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, and frame rate***

1. Introduction

Controller Area Network (CAN) in real time applications aims 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 set-up. The CAN controller features allow the efficient implementation of higher level protocols without affecting the performance of the microcontroller. The CAN or the CAN bus itself is a balanced interface bus where the complementary signals are sent over two wires, and the voltage difference between the two wires determines the logical state of the bus. The differential CAN receiver oversees this voltage difference and outputs the bus state with a single-ended output signal [1].

The Bit Encoding scheme used in CAN protocol is Non Return to Zero (NRZ) line encoding for data communication and ensuring compact messages with a minimum number of transitions and high adaptability on external disturbances. However, due to insufficient signal edges for synchronization of bit stream, bit stuffing is required [2]. Bit stuffing guarantees the receiver in synchronous state, especially when the data stream sent a large volume of data in a same polarity, either dominant or recessive manner that has no transition of the signal. Also 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 or recessive, within these frames is considered an error condition [3]. 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 number of stuffed bits for both frames.

$$stuff\quad bits\quad = [\frac{34+8*(l)}{5}] \qquad (1)$$

where *l* is the data length in bits and it is measures 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 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 [4]. 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.

The rest of this paper is organized as follows. Section 2, describes some existing line encoding. Then, section 3 briefly discuss the CAN Frame Rate. While 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. Principle of line encoding techniques

Sending binary data through a network requires that the physical quantities' characteristics 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. Also 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 un-coded signal. Error detection and correction capability is one of the important factors because the line code should have an error detection and an error correction capabilities.

This work was supported by the IT and R&D Program of Ministry of Trade, Industry and Energy (10049192, Development of Smart Automotive ADAS SW-SoC for Self-Driving Car).

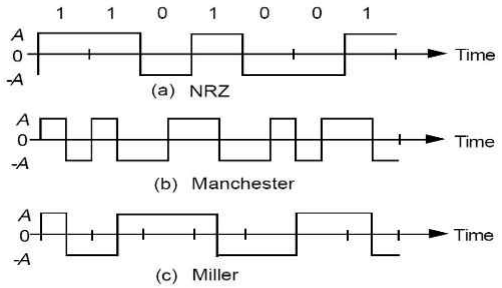

Figure 1. Signal waveform of a) NRZ, b) Manchester and c) Miller.

2.1 Polar Non Return to Zero (NRZ) Code

A polar NRZ signal is also called a NRZ-L (L for 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 shows 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 raising 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 [5]. The signal waveform of this line encoding is shown in Figure 1 (b).

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 [6]. 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.

3. CAN Frame rate

CAN implements bit-stuffing protocol that no more than five consecutives 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 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 [7], the CAN bus frame rate is defined by a given baud rate divided by the total number of frame bits and measures in frames per second as shown in Equation (2).

$$frame\ rate\quad (FR) = \frac{B_R}{T_{FB}} \qquad (2)$$

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.

		Arbitration Field		Control Field			Data Field	CRC		ACK			
	SOF	Identifier	RTR	IDE	r0	DLC		CRC Sequence	CRC Delimiter	ACK slot flag/bit	ACK 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	1111111	111

Legend for Assigned bits:

1 Recessive bit
0 Dominant bit
X Bit may be either state

SOF - Start Of Frame
RTR - Remote Transmission Request
IDE - Identifier Extension
r0 - Reserved Bits
DLC - Data Length Code

CRC - Cyclic Redundancy Check
ACK - Acknowledge
EOF - End OF Frame
IFS - Inter Frame Space

Figure 2. Standard CAN Data Frame (2.0 A Frame).

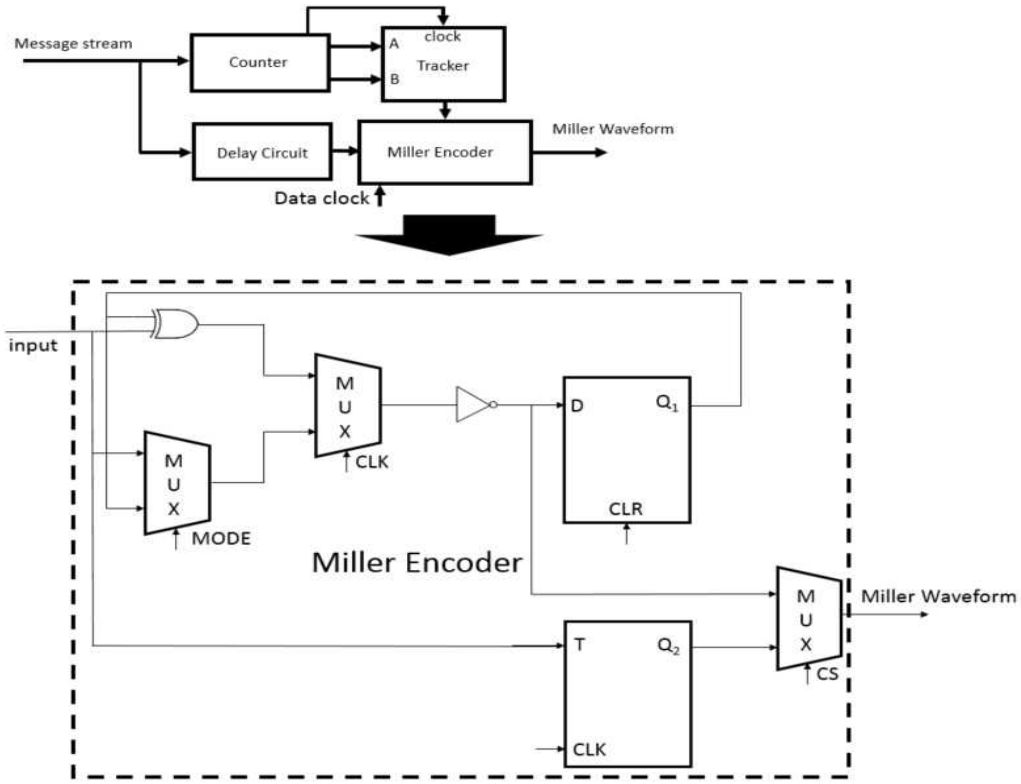

Figure 3. Proposed Architecture of Miller Line Encoding.

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. The data stream is input to a delay circuit and a counter. The counter counts the number of zeros and ones and produces signal indicating whether the number of 1s and 0s is odd or even. The counter also supplies clock signal to tracker at a clock input terminal. The output signal from the delay circuit is applied as an input to a Miller encoder. The information is enclosed in the transitions of the signal level.

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

The proposed implementation of this algorithm is describe as follows:

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

- A “1” causes a transition from one level to the other one in the middle of the bit period.
- A “0” following a “1” no transition.
- A “0” following a “0” causes a transition to the other level at the beginning of the bit period.

5. Experimental Results

The proposed implementation was synthesized to Xilinx Virtex 5 FPGA by using Xilinx ISE. Table 1 shows the device utilization result from the Xilinx synthesis tool. Figure 4 shows the simulation results in using Verilog HDL, the input “a_in” is 1 and the clock is given as a rising edge which produces the output as 1. If Miller line encoding is implemented, stuffed bits can be eliminated, Table 2 the comparison of Frame length variations between NRZ and Miller line encoding both for frames 2.0 A and 2.0 B.

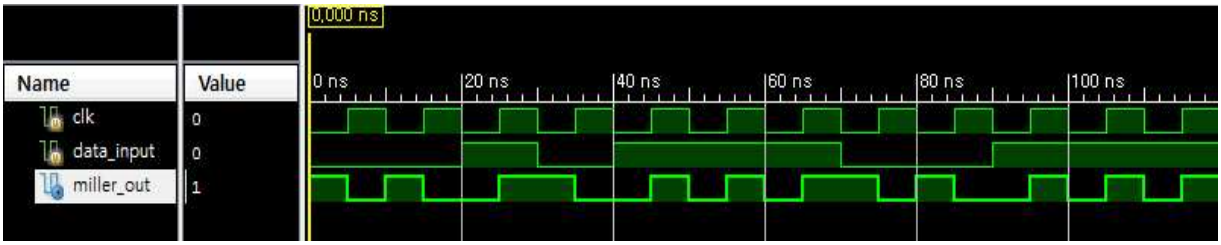

Figure 4. Simulation result of Miller encoding.

Table 1. FPGA Device Utilization Summary

Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	1	12,480	0%
Number of fully used LUT-FF pairs	0	1	0%
Number of bonded IOBs	3	172	1%

Table 2. Frame Length Variation between NRZ and Miller Line Encoding

	NRZ		Miller	
	2.0 A Frame	**2.0 B Frame**	**2.0 A Frame**	**2.0 B Frame**
	Date Field 8 bytes	Date Field 8 bytes	Date Field 8 bytes	Date 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 3 shows the Miller Line Encoding minimizes the frame payload and resulting an increase in frame rate of the controller, both frames 2.0 A and 2.0 B uses 125 kbps of baud rate at 8 bytes bits data.

6. Conclusion

A 100% error free transmission cannot be attained in real time especially if the transition density on the data sequence is in a long runs of 0s or 1s. Miller Line Encoding benefits the elimination of bit stuffing process that causes jitter in CAN communication and also leads with the reduction of frame length. This proposed paper achieved the aim of increasing the frame rate of CAN controller as it was shown in the simulation results. Both Tables 2 and 3 shows that the performance of Miller Line Encoding is much better than NRZ in reduction of frame length and it is much better in terms of frame rate. Also the system's payload decreases and the required stuff bits was minimized.

Table 3. Frame Rate comparison using NRZ and Manchester both for Frames 2.0A and 2.0B

	NRZ	Miller
Frame	**Frame rate (frame per second)**	**Frame rate (frame per second)**
2.0A	**984**	**1,126**
2.0B	**811**	**954**

References

[1] http://www.esa.int/Our_Activities/Space_Engineering_Technology/Onboard_Computer_and_Data_Handling/CAN_-_Controller_Area_Network_Bus
[2] W. Voss, "Controller Area Network, Serial Network Technology for Embedded Solutions", ESD Electronics, pp. 29-30, 2008.
[3] W. Voss, "A comprehensive Guide to Controller Area Network", ESD Electronics, pp. 96, 2008.
[4] Numgi Kim, "Variable CRC Scheme for Efficient Data Transmission Control for IEEE 802.16e Wireless Network", JKIIT, Vol. 7, No. 3, pp. 109-115, July 2009.
[5] H. Bidgoli, "Handbook of Computer Networks: Key Concepts, Data Transmission, and Digital and Optical Networks", pp. 493, 2008.
[6] W. C. Lindsey and M. K. Simon, Telecommunication Systems Engineering, Prentice-Hall, 1973.
[7] http://electronics.stackexchange.com/questions/121329/whats-the-maximum-can-bus-frame-message-rate-at125-kbit-s