International Journal of Pure and Applied Mathematics
Volume 119 No. 14 2018, 1739-1745
ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
Special Issue

ijpam.eu

COMPARISON OF CAN AND FLEXRAY PROTOCOL FOR AUTOMOTIVE APPLICATION

J.Pradeep[1], S. Richerd Sebasteen[2], R. Dineshkrishna[3]
[1]Department of ECE, [3]Dept of Mechanical Engineering, Sri SMVC, Puducherry, India
[2]Department of EIE, SJCET, Chennai, India

ABSTRACT— Today for a security critical and safety application in automotive industry, a protocol is required to transfer the data with a higher speed than existing protocol. As concepts like steering arms and brake pipes are replaced the electric wire and integrated circuits, a robust and fail safe communication is needed, and FlexRay is fulfilling these needs. CAN (Controller Area Network) is an existing multi-master broadcast serial bus communication protocol for connecting Electronic Control Units in automotive application, FlexRay is a newly introduced, communication protocol for automotive control system, which is developed to fulfill the increasing demands in automotive for higher safety and comfort. Flex Ray can overcome the disadvantage of existing CAN protocol. This paper provides a comparison of the leading communication protocols for high performance applications. The goal of this paper is to provide a valuable resource when comparing and selecting communication protocols for these critical vehicle applications. The protocols are analyzed for comparison. It is designed and demonstrated for the both frame structure. The simulation and Synthesis result are presented in this paper using Xilinx software as tool

Keywords— Electronic Control Unit (ECU), Controller Area Network (CAN), FlexRay (FR)

INTRODUCTION

Nowdays, automotive system are complex distributed embedded system to compose several ECU interconnected by communication network. For non-safety critical application, a number of popular protocols are available such as LIN, CAN, Bluetooth, and ZigBee. The Controller Area Network (CAN) is a serial communications protocol which efficiently supports distributed real time control system with a very high level of security but it does not met the set of challenging conditions & requirements. In general, CAN protocol is based on event-driven communication approach, because each bus node of a communication system can able to access common communication medium with data rate of 1 Mbit/sec [1]. For high data rate, the CAN bus did not meet the requirements of fault tolerance. Mainly for safety critical and more complex application, several communication protocols have been introduced such as Byteflight, TTP/C, TT-CAN, and FlexRay. In the automotive industry the trend is using the mixed pattern, with the FlexRay protocol being pointed out as a potential de facto standard for high speed communication in cars.

Normally, the data exchange between the numerous controls devices, sensors, actuators in automobile is carried out via CAN network. However the introduction of the new X-by-wire systems results in increased requirements especially with regards to error tolerance and time determinism of message transmission. FlexRay fulfills these requirements by message transmission in fixed time node and fault tolerance with redundant message transmission on two channels.

The literature review of CAN and FlexRay protocol is as follows. In the early 1980s, Robert Bosch GmbH Company has developed a new network controller in the automotive industries with the intention of replacing and simplifying the wiring system [2]. The CAN system was developed as a serial bus with high speed, high reliability, and low cost for distributed real time control applications [3]. CAN is a de facto standard for data transmission in automotive applications [4]. Slowly, CAN begin to gain wide appreciation in various industrial automation, due to its high immunity towards electrical interference and the ability to self-diagnose with repair data errors [5]. With its low cost, high performances, upgradeability, and providing flexibility, researchers initiated to implementing this protocol in military, aviation, electronics, factories and many other industries [6]. Flex-Ray will emerge as the predominant protocol for in-vehicle automotive communication systems. As a result, lot of recent interest in timing and predictability analysis techniques are specifically targeted towards Flex-Ray. In this paper [7] authors have proposed a compositional performance analysis framework for a network of electronic control units (ECUs) that communicate via a Flex-Ray bus. In contrast to previous timing analysis techniques which

analyze the Flex-Ray bus in isolation, the framework is fully compositional and allows the modeling of the schedulers at the ECUs and the Flex-Ray protocol in a seamless manner. Flex-Ray become the de-facto standard for in-vehicle communications. Its main advantage is the combination of high speed static and dynamic transmission of messages. Authors have proposed techniques for optimizing the Flex- Ray bus access mechanism of a distributed system, hard real-time deadlines are met for all the tasks and messages in the system.

Rest of this paper is organized as follows. In section II describes the background about the two protocols. In section III presents the overview of two protocols frame structure. In section IV presents VHDL simulation and synthesis results with comparion. Finally it concludes the paper in section V.

I. BACKGROUND

A. CAN Protocol

CAN is one of the most advanced serial communication protocols used in applications. In automotive applications, CAN is widely used as a communication protocol for ECUs due to its enhanced features. As a communication protocol, CAN complies with the two bottom layers of the Open Systems Interconnection model (OSI) standardized by the International Standards Organization (ISO). CAN communication controller is used to control communication between devices connected on the bus. CAN use an arbitration feature to control access of devices connected on the same bus, in order to avoid transmission collision, which causes errors in communication. Although CAN has features like tolerate communication faults, permanent and intermittent faults that could not be detected and confined by the error management unit may occur on the bus. Faults could come from any of the parts that form the communication links. A CAN device that uses 11 bit identifiers is commonly called CAN 2.0A and a CAN device that uses 29 bit identifiers is commonly called CAN 2.0B. These standards are freely available from Bosch along with other specifications and white papers. In 1993 the International Organization for Standardization released the CAN standard ISO 11898 which was later restructured into two parts; ISO 11898-1 which covers the data link layer, and ISO 11898-2 which covers the CAN physical layer for high-speed CAN. ISO 11898-3 was released later and covers the CAN physical layer for low-speed, fault-tolerant CAN. The physical layer standards ISO 11898-2 and ISO 11898-3 are not part of the Bosch CAN 2.0 specification [2].

The CAN specifications use the terms dominant bits and recessive bits where dominant is a logical 0 (actively driven to a voltage by the transmitter) and recessive is a logical 1 (passively returned to a voltage by a resistor). The idle state is represented by the recessive level (Logical 1). If one node transmits a dominant bit and another node transmits a recessive bit then there is a collision and the dominant bit wins. The means there is no delay to the higher priority message, and the node transmitting the lower priority message automatically attempts to re-transmit six bit clocks after the end of the dominant message. This makes CAN very suitable as a real time prioritized communications system.

Bit rates up to 1 Mbit/s are possible at network lengths below 40 m. Decreasing the bit rate allows longer network distances (e.g., 500 m at 125 kbit/s). The improved CAN FD standard allows increasing the bit rate after arbitration and can increase the speed of the data section by a factor of up to eight of the arbitration bit rate.

B. FlexRay Protocol

FlexRay is an automotive network communications protocol developed through the FlexRay Consortium to govern on-board automotive computing. It is designed to be faster and reliable than CAN and TTP, but it is more expensive. Visual Sim FlexRay library enables a system designer to construct models of complex standard and non-standard FlexRay topologies. The graphical model of the FlexRay topology can contains any number of nodes that transmit across the static and dynamic slots. The FlexRay consortium disbanded in 2009. The FlexRay standard is a set of ISO standards ISO 17458-1 to 17458-5.

FlexRay supports high data rates, up to 10 Mbit/s, explicitly supports both star and party line bus topologies, and it consist of two independent data channels for fault-tolerance.The bus operates on a time cycle, divided into two parts: the static segment and the dynamic segment. The static segment is pre-allocated into slices for individual communication types, providing a stronger real-time guarantee than its predecessor CAN. The dynamic segment operates more like CAN, with nodes taking control of the bus as available, allowing event-triggered behavior.

FlexRay protocol is the pre-dominant protocol. It has several advantages in terms of bandwidth, message transmission and fault tolerant mechanism. The FlexRay development is driven through leading core companies such as BMW, Daimler Chrysler, Freescale

semiconductor, GM, NXP semiconductors, Robert Bosch, and Volkswagen.

A FlexRay communication system consists of number of FR nodes and physical transmission medium (FR bus) for interconnecting all of the FlexRay nodes. FR nodes are also called as ECU, which is connected to a FlexRay bus via a FR interface. FlexRay interface consist of communication Controller and one/two bus drivers depending up on the number of channels. Basically in FR communication system, as two channels for designing it can choose between single channel and dual channel configuration. Based on the design the communication channels operate at the data rate of 10 Mbit/sec [7].

FR protocol supports the operation of a communication controller with single/redundant communication channel. In case of single communication channel configuration, all controllers are attached to the communication channel via one port. In case of redundant configuration controller, it is attached to the communication via more than one port. Controllers connected to two channels are configured to transmit data redundantly on two channels at the same time. This redundant transmission allows the masking of a temporary fault of one communication channel and it constitutes a powerful fault tolerance feature of the protocol. A second fault tolerance related to transients fault constructed by the redundant transmission of data over the same channel with particular time and delay between the redundant transmissions. The delayed transmission allows tolerating transients fault on both channels under particular pre-condition. If fault tolerant property of two independent channels is not required for specific application, the channel is used to transfer the different data with effectively doubling the transmission bandwidth.

II. OVERVIEW OF CAN AND FLEXRAY FRAME STRUCTURE

A. Can Frame Structure

A CAN network is configured to work in two different frame formats: the standard/base frame format (described in CAN 2.0 A and CAN 2.0 B), and the extended frame format (only described by CAN 2.0 B) is shown in Figure.1. The difference between the two formats is that, the CAN base frame supports a length of 11 bits for the identifier, and the CAN extended frame supports a length of 29 bits for the identifier, made up of the 11-bit identifier (base identifier) and an 18-bit extension (identifier extension). CAN controllers that support extended frame format messages are also able to send and receive messages in CAN base frame format. CAN have four frame types;

(a) Data frame is containing node data for transmission
(b) Remote frame is used to requesting the transmission of a specific identifier
(c) Error frame transmitted by any node detecting an error
(d) Overload frame to inject a delay between data and/or remote frame

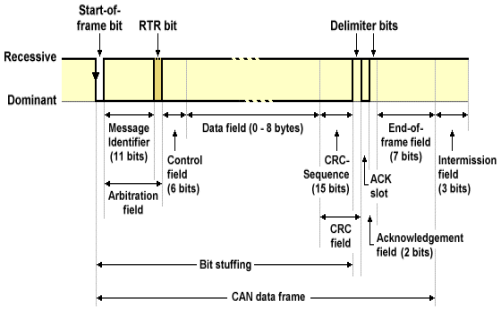

Figure.1 Frame transmission in CAN

The data frame is the only frame for actual data transmission. There are two message formats, base frame format with 11 identifier bits and extended frame format with 29 identifier bits. In remote frame the data transmission is performed on an autonomous basis with the data source node (e.g., a sensor) sending out a data frame. It is also possible, however, for a destination node to request the data from the source by sending a remote frame. There are two differences between a data frame and a remote frame. Firstly the RTR-bit is transmitted as a dominant bit in the data frame and secondly in the remote frame there is no data field.

- RTR = 0 ; DOMINANT in data frame
- RTR = 1 ; RECESSIVE in remote frame

In the unlikely event of a data frame and a remote frame with the same identifier being transmitted at the same time, the data frame wins arbitration due to the dominant RTR bit following the identifier. In this way, the node that transmitted the remote frame receives the desired data immediately.

The error frame consists of two different fields. The first field is given by the superposition of ERROR FLAGS (6–12 dominant/recessive bits) contributed from

different stations. The following second field is the error delimiter (8 recessive bits). There are two types of error flags:

- Active Error Flag: Six dominant bits – Transmitted by a node detecting an error on the network that is in error state (error active).
- Passive Error Flag: Six recessive bits – Transmitted by a node detecting an active error frame on the network that is in error state (error passive).

The overload frame contains the two bit fields Overload Flag and Overload Delimiter. There are two kinds of overload conditions that can lead to the transmission of an overload flag, the internal conditions of a receiver, which requires a delay of the next data frame or remote frame and detection of a dominant bit during intermission. The overload flags form destroys the fixed form of the intermission field. As a consequence, all other stations also detect an overload condition and on their part start transmission of an overload flag. Overload delimiter consists of eight recessive bits. The overload delimiter is of the same form as the error delimiter [4].

B. FlexRay Frame Format

An overview of the general FlexRay frame format is illustrated in Figure.2. The FlexRay frame format divides into three segments: Header, Payload, and Trailer.

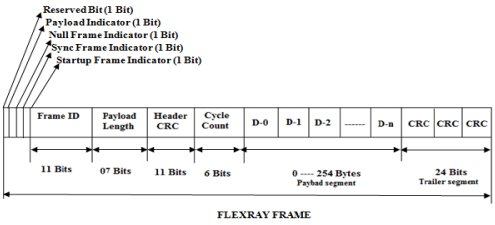

Figure.2 FlexRay Frame Format

C. Header:

The FlexRay header segment is 5 bytes long. It consists of 9 parts. Each part has the different function.

a. Reserved bit (1 bit) – It is reserved for future protocol use

b. Payload preamble indicator (1 bit) – It indicates the extension of vector information in the frames payload segment. If the frame is transmitted in the static segment, this position indicates the presence of a network management vector at the beginning of the payload. If the frame is transmitted in the dynamic segment, this position indicates the presence of a frame ID at the beginning of the payload.

c. Null frame indicator (1 bit) – It is to identify the frame is empty. ”0” means the payload segment contains no valid data; “1” means the payload segment contains valid data.

d. Sync frame indicator (1 bit) – It is to check the frame is a sync frame. If “0” means no synchronization for node; and “1” means all receiving nodes are used for synchronization

e. Startup frame indicator (1 bit) – It indicates frame is a startup frame. “0” means this frame is not a startup frame; “1” means this frame is a startup frame.

f. Frame ID (11 bits) - A frame ID is unique on each channel in CC. The frame ID ranges from 1 to 2047. 0 is invalid.

g. Payload length (7 bits) - This part is used to indicate the size of the payload segment. The value of the payload length position is set to the number of payload bytes divided by 2. Its range is from 0 to 254 bytes.

h. Header CRC (11 bits) - This part contains a cyclic redundancy check code (CRC), the startup frame indicator, the frame ID, and the payload length. The header CRC of transmitted frames is computed offline and provided to the Communication Controller by means of configuration. It is not computed by transmitting CC. The CC calculates the received frame’s header CRC in order to check that the CRC is correct.

i. Cycle count (6 bits) - This part indicates the value of cycle counter, from the transmitting node's at the time of frame transmissions.

D. Payload:

The FlexRay payload segment contains 0 to 254 bytes data (0 to 127 two-byte words). It is used to notice that the payload segment contains even number of bytes

E. Trailer:

The FlexRay trailer segment consists of 24-bit cyclic redundancy check code (CRC) for the frame. It is computed with the data in the header segment and the payload segment of the frame [7].

III. SIMULATION AND SYNTHESIS RESULT

To show the comparison result of CAN and FlexRay protocol the frame structure is designed using Verilog

code and Xilinx tool (version 14.1) are shown below. In Figure.3 it shows the design of FlexRay frame structure. The output of frame depends on input command (4-0), CRC and reset signal. The FlexRay frame is divided into three segments. The segments are Header, Payload, and Trailer. Commands apply by Host for respective state ("00000" to "10001"). Figure 3 & 4 shows the design of proposed FlexRay frame structure and its RTL viewer.

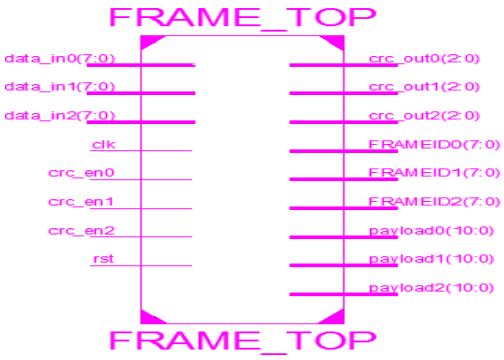

Figure.3 RTL Viewer of proposed FlexRay Frame structure

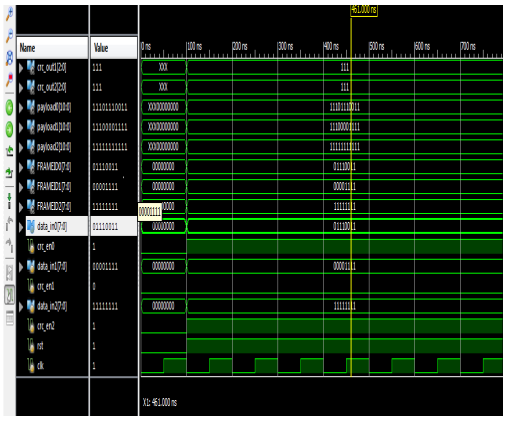

Figure.4 Design of proposed FlexRay Frame structure

In Figure.5 & 6 it shows the frame structure of CAN protocol. The output of frame consists of data frame, cyclic redundancy check, remote frame, error frame, and overload frame.

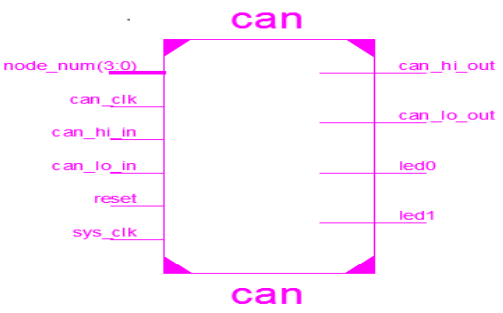

Figure.5 RTL Viewer of CAN Frame structure

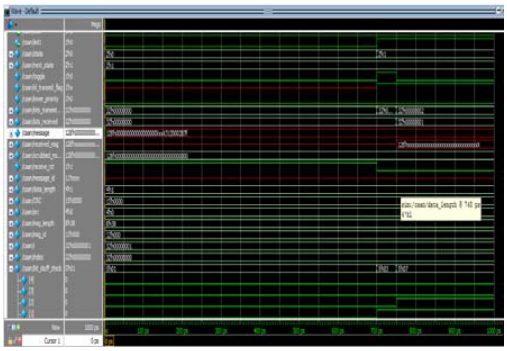

Figure.6 Design of CAN Frame structure

Table.1 shows the summary value of CAN and Flexray frame structure. From the summary value it defines the logic utilization of slice register; Look up Table, bounded IOB's used for both CAN and FlexRay protocol.

Table.1 Summary of CAN and FlexRay

Logic utilization	Available	CAN	Flexray
Number of slice registers	126800	15	9
Number of LUTs	63400	15	9
Number of Bonded IOBs	210	110	95
Number of BUFG	32	1	1

In Table.2 it shows the comparison value of CAN and FlexRay frame structure. From the comparison value it shows the FlexRay protocol having more advantages in

automotive systems comparing to CAN protocol. The values obtain from clock period and delay for CAN protocol is high comparing to FlexRay protocol, so that the bit value sending through CAN frame structure cause more delay and usage of clock is high compare to FlexRay protocol.

Table.2 Comparison of CAN and FlexRay

Protocols	Minimum Clock Period	Maximum Path Delay	Maximum Frequency
CAN	1.208 ns	1.08 ns	1596.67 MHz
FlexRay	0.771 ns	0.290 ns	1296.849 MHz

IV. CONCLUSION

In this paper, The FlexRay with widely used CAN protocols are analyzed for comparison is highlighted. CAN protocol have small data package with poor determinism and lack of suitable network scheme led to the development of FlexRay protocol. From the comparison value of CAN and FlexRay frame structure, clock period value is high for CAN protocol compares to FlexRay protocol and its delay value is reduce for 0.79 ns in FlexRay protocol.

REFERENCES

[1] Avinash K R,P Nagaraju, Surendra S,Shivaprasad S, "FlexRay protocol based an automotive application", International journal of Emerging technology and Advanced Engineering (IJETAE),vol.2,no.5,May 2012.

[2] Robert Bosch GmbH, "CAN Specification", Version 2.0, 1991.

[3] Ekiz, H.Kutlu, A. Powner, E.T,"Implementation of CAN / CAN Bridges in Distributed Environments and Performance Analysis of Bridged Can Systems Using Sae Benchmark, Proceedings of IEEE Southeastcon, vol.5, pp.185 – 187, Aug.1997

[4] Navet,"Controller area network [automotive applications]", IEEE Potentials, vol. 4, no. 4, pp. 12 – 14, Nov.1998.

[5] Alheraish,"Design and Implementation of Home Automation System", IEEE Transactions on Consumer Electronics, vol. 50, no. 4, pp. 1087 – 1092, May 2004.

[6] Kumar, M. A.Verma, and A. Srividya,"Modeling of Controller Area Network (CAN)" Distributed Computing and Networking, Lecture Notes in Computer Science vol.5408, p 163-174, June 2009.

[7] Flexray Consortium: FlexRay Requirements Specification (Version 2.1), December 2005.URL http://www.flexray.com/.

[8] Wei Lun Ng, Chee Kyun Ng, Borhanuddin Mohd. Ali, Nor Kamariah Noordin, and Fakhrul Zaman Rokhani, "Review of Researches in Controller Area Networks Evolution and Applications", 2nd conference in Information and Communication Technologies, vol. 2, pp. 3088 – 3093, March 2006.

[9] Shanker Shreejith Student Member, IEEE and Suhaib A. Fahmy Senior Member "Extensible FlexRay Communication Controller for FPGA-Based Automotive Systems" IEEE Transacation on Vehicular technology, vol.6, no.99, May 2014.

[10] B. Kim and K. Park, "Probabilistic Delay Model of Dynamic Message Frame in FlexRay Protocol," IEEE Transaction on Consumer Electronics, vol. 55, No.1, pp. 77–82, 2009.

[11] N. Xu, Y. E. Kim, K. J. Cho, J. G. Chung, and M. S. Lim, "Implementation of FlexRay Communication Controller Protocol with Application to a Robot System," Proceeding of IEEE International Conference on Electronics, Circuits and Systems, vol.9, pp.994-997, Sep. 2008.

[12] A. Albert, "Comparison of event-triggered and time-triggered concepts with regard to distributed control systems," Embedded world,vol.4,pp.235-252,Feb.2004.

[13] Milind Khanapurkar,Jayant Y,Hande and Dr.Preeti Bajaj,"Approach for VHDL and FPGA implementation of communication controller of FlexRay controller", Proceeding of Journal of International Hiding and Multimedia signal processing,vol.1,no.4,Oct.2010.

[14] S. Shreejith, K. Vipin, S. A. Fahmy, and M. Lukasiewycz, "An Approach for Redundancy in FlexRay Networks Using FPGA Partial Reconfiguration," Proceeding of Design, Automation and Test in Europe Conference, pp. 721–724, Mar.2013.

[15] S. Chakraborty, M. Lukasiewycz, C. Buckl, S. Fahmy, N. Chang, S. Park, Y.Kim, P. Leteinturier, and H. Adlkofer, "Embedded Systems and Software Challenges in Electric Vehicles," Proceeding of Design, Automation and Test in Europe Conference, pp.424-429, Mar.2012.