© 2016 IJSRSET | Volume 2 | Issue 1 | Print ISSN : 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Flexray Communication Controller for Intra-Vehicular Communication and Its Realization in FPGA

Anita K. Joseph, Prof. G. K. Sadanandan

Department of Electronics and Communication Engineering, Toc H Institute of Science and Technology, Kerala, India

ABSTRACT

Intra-vehicular communication describes an exchange of data within the ECUs (Electronic Control Units) of the vehicle which are involved in vehicular applications. The Flex Ray protocol is a unique time-triggered protocol that provides options for deterministic data that arrives in an expectable time frame. In case of CAN (Controller Area Network) it provides a dynamic event-driven data to handle a large variety of frames. The Flex Ray communication cycle is the fundamental element of the media-access scheme within Flex Ray. The main advantage of flex Ray communication protocol is its constant latency. It also have dual channel for redundancy or faster transfer. This paper describes a field-programmable gate array (FPGA)-based communication controller (CC). This features configurable extensions to provide functionality that is unavailable with standard implementations or off-the-shelf devices. VHDL simulation & FPGA synthesis results of communication controller of flexray controller are tested on Quartus II version 12.1 are presented.

Keywords: Flexray, Electronic Control units, Field Programmable Gate Array

I. INTRODUCTION

Today modern cars contain a multiple of controllers that are networked together by various bus communication systems with very different properties. Automotive communication networks have access to several crucial components of the vehicle, like breaks, airbags, and engine control. Moreover, cars that are equipped with driving aid systems like ESC (electronic stability control) or ACC (adaptive cruise control) allow deep interventions in the driving behavior of the vehicle. Further electronic drive-by-wire vehicle control systems will fully depend on the underlying automotive data networks.

For example, the engine needs to tell the transmission what the engine speed is, and the transmission needs to tell other modules when a gear shift occurs. This need to exchange data quickly and reliably led to the development of the vehicle network. This vehicular network is the medium of data exchange between the ECUs in the automotive systems.

Today's control and communications networks are based on serial protocols. So it reduces the total wire length used to interconnect ECUs. Moreover it counters the problem of large amounts of discrete wiring. One of the main objectives of the design step of an in-vehicle embedded system is to ensure a proper execution of the vehicle functions with a predefined level of safety. Networks play a central role in maintaining the electronic control systems working properly, because most of the core elements are distributed and need to communicate with each other. As a result, there is a need for designing different automotive networks capable of meeting different applications' requirements. Such as, Local Interconnected Network (LIN) used for transmitting simple control data with data rates lower than 10kb/s. The Low-speed Controller Area Network (CAN) designed for data sharing and exchanges between sensors and ECUs. The High-speed CAN designs for high speed real-time communications [1]. The Media Oriented System Transport (MOST) network designs for the multimedia data which has high data rates. New

applications like x-by-wire need new features from the control networks such as predictability, fault tolerance and flexibility. These motivate the development of new automotive control networks, for example, Time Triggered Protocol (TTP) and Flex Ray [2]. Furthermore, in order to adapt different in-vehicle applications, the control networks should be able to support both time-triggered (TT) and event-triggered (ET) transmission. This type of network is Flex Ray.

The remainder of this paper is organized as follows. Section II details the controller architecture along with the flexray frame format and protocol operation control. Simulation results and synthesized reports are provided in section III. Finally, we conclude this paper in section IV.

II. METHODS AND MATERIAL

Flexray Communication Controller

The FlexRay communication protocol is specified for a reliable automotive network. The protocol was industrialized for providing fault-free communication between various electronic components of communication system.

A system which consists of finite quantity of electronic components capable of exchanging data is named as distributed system and also calls the electronic control units as nodes. A distributed system in which nodes are connected via at least one communication channel directly in like bus topology or by star couplers in like star topology is called cluster.

The FlexRay communication protocol uses TDMA to the media for providing communication between nodes. It means that every node must have at least one unique predefined slot of time [3]. During this interval of time the node has access to the media for sending its data. To send the data to the media, there is a structure used by the communication system, we shall call this structure as communication frame and it is depicted in figure1.

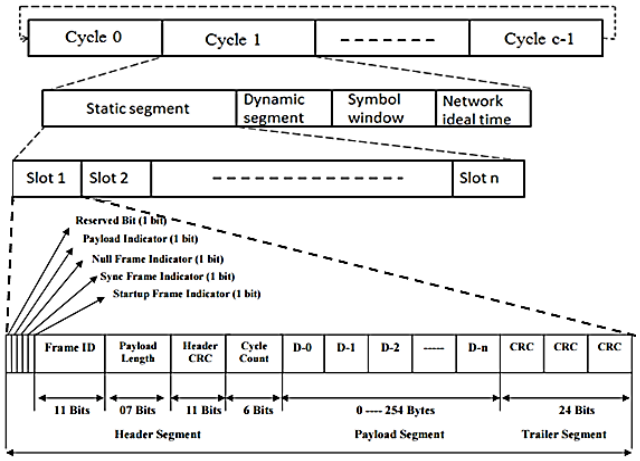

Figure 1 : Flexray Communication Cycle

A. Architecture of FlexRay

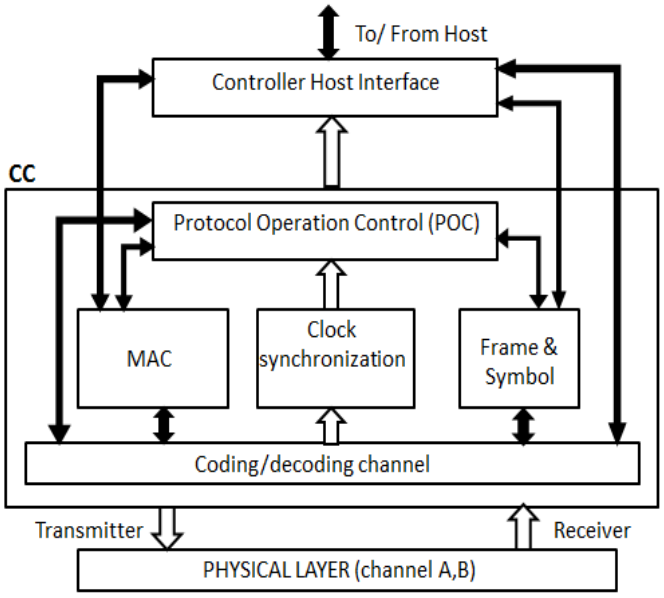

Figure 2 : FlexRay node

Figure 2 shows a generic structure of a FlexRay node. Node generally has two parts, the controller part and the driver part. The controller consists of a Communication Controller (CC) and a Host CPU. The driver part consists of a Bus Driver, optional to have a Bus Guardian.

1) Host CPU:

The Host CPU is a part of an ECU where the application software is executed. The Host CPU provides the control and configuration information to the CC, also provides payload data transmitted during the communication cycle.

2) Communication Controller (CC):

The Communication controller is an electronic component in a node that is responsible for implementing the protocol aspects of the FlexRay communications system [4]. It provides status information to the host and delivers payload data received from communication frames.

3) Bus Driver (BD):

Bus driver is an electronic component consisting of a transmitter and a receiver that connects a communication controller to one communication channel. It can be used in electrical encoding/decoding, remote wake-up and error detection on OSI layer 0 (voltage, temperature).

4) Bus Guardian (optional):

Bus Guardian protects a channel from interference caused by communication that is not temporally be scheduled within limited of the times in a schedule. Namely, it restricts transmissions of CC to defined slots, fault containment for fail-safe node communication, supports error detection and fault tolerance. The main processes of a node accesses to the bus are as following; BD first connects the CC and the bus. The BG monitors the connection which accesses to the bus. The Host CPU informs the BG which time slots are allocated by the CC. The BG only allows CC to transfer data at these time slots. It also activates the BD. If the BG detects an idle interval in time, then it disconnects the node with the communication channel.

B. Protocol Operation Control (POC)

POC is the heart of the communication controller. It operates based on a state machine. This controls and executes the states of the whole system. At the figure 3, we depict the states of the protocol operational control.

The node enters the default config state when the node is supplied with power. In this state the default static configuration of the communication controller is performed. Here host configures number of channels, media specification, etc. This state is left as soon as the bus driver receives a wakeup event which can be an external wakeup event or receiving the wakeup pattern on its channel.

In the config state, the host configures the CC with dynamic configuration data. Receiving wakeup pattern from another node host checks what kind of wakeup event is received (if it was external or internal) and based on this information decides what state will be next. If the bus driver has received an external wakeup event, the host must command to wakeup other nodes on an attached channel. Otherwise, if the bus driver has received a wakeup pattern from the channel, the host may command to wake up the second channel (if the node is attached to both channels), or to proceed to startup state [5].

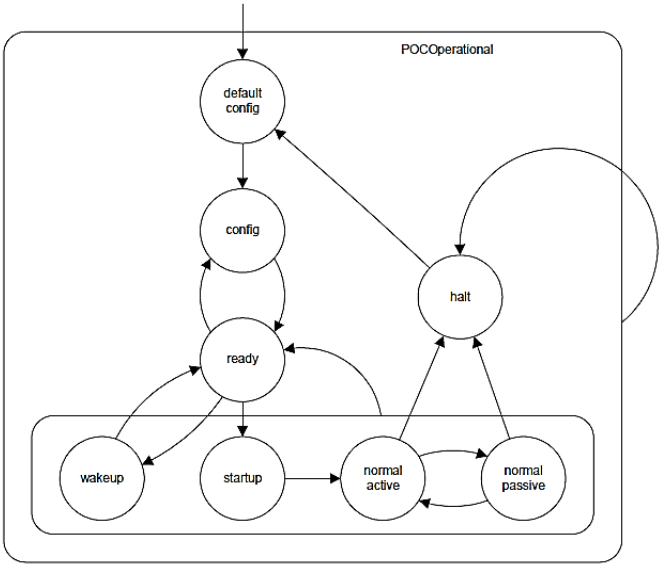

Figure 3 : Protocol operation control

In the wakeup state, the flexray node tries to wake up the defined channel. If any error occurred during the wakeup state, the CC informs the host and the node switches to the config state. Otherwise the node switches to startup state [6].

In the startup state, the node checks for ongoing communication on the media. If the bus driver detects any communication signals on the attached channel, then the node integrates to the cluster communication, otherwise the node starts the startup procedure [7]. After startup, node proceeds to the normal active state in this state it performs the normal error check. In the passive state node checks the synchronization error. If errors occurred, the node comes to config state though the halt state and tries the whole procedure again.

III. RESULTS AND DISCUSSION

SIMULATION RESULT

These sections detail the simulation result and synthesize report of the flexray communication controller. The designed flexray controller input frames arrived from host CPU to the CC. We have kept frame size of 254 bits. This includes 8 bytes of data frame. The payload data is the actual data which is to be communicated. The CC of the node modifies the input data according to the flexray specifications. And allow communication through the specified channel. Here figure 4 represents the transmitted data and figure 5 represents the received data. Table 1 shows the analysis of transmitted data and receiving data along with its corresponding state changes.

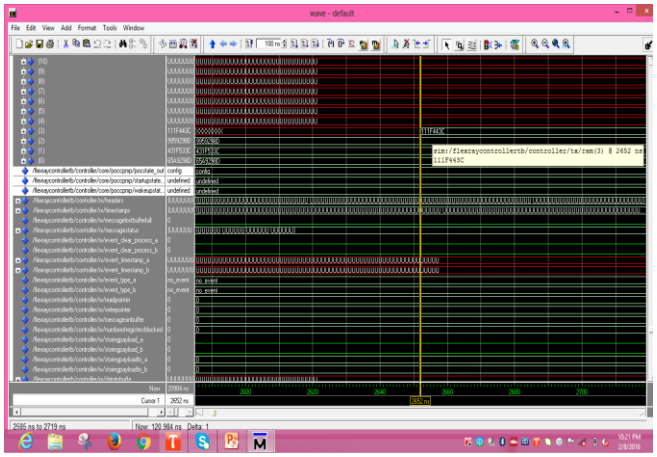

Figure 4 : Transmitted data

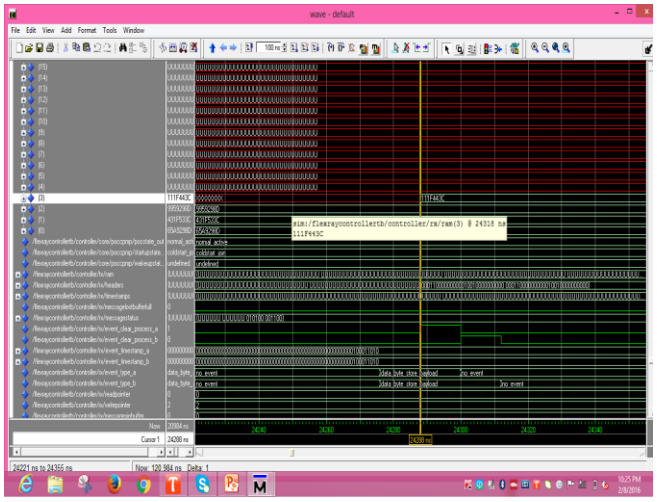

Figure 5 : Received data

TABLE 1: ANALYSIS TABLE

POC state during data transmiss-ion	Transmitt-ed Data's	POC state during receiving	Received data's
Config state	65A9298D	Normal active	65A9298D
Config state	431F533C	Normal active	431F533C
Config state	9959298D	Normal active	9959298D
Config state	111F443C	Normal active	111F443C

SYNTHESIS RESULTS

RTL Schematics

RTL Schematic is the output of synthesis tool corresponding to the code that is being synthesized. It is pictorial representation of the synthesized circuit. Figure 6 represents the RTL view of the flexray communication controller.

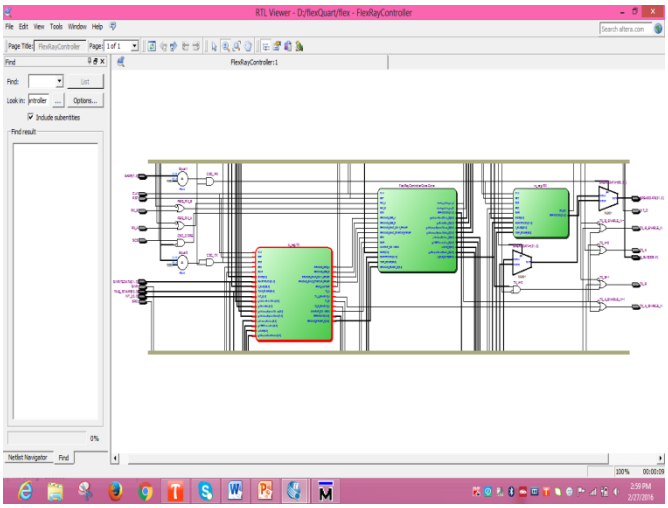

Figure 6 : RTL view of the Flexray Communication controller

Design Summery

The design of flexray communication controller in VHDL using Quartus II has the following results depicted in table 2.

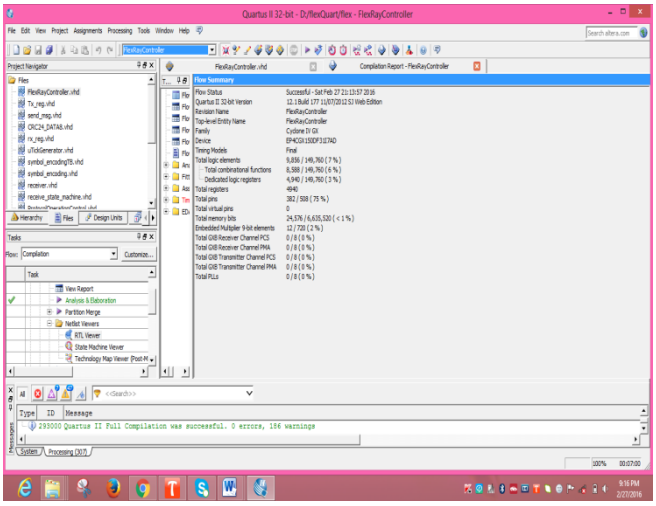

Figure 7: design summery

TABLE 2: DEVICE SUMMARY OF FLEXRAY COMMUNICATION CONTROLLER

Parameters	Used	Available
Total logic elements	9856	149760
Total combinational functions	8588	149760
Total registers	4940	149760
Total pins	382	508
Total memory bits	24576	6635520

IV.CONCLUSION

The flexray communication controller for intra-vehicular communication has been designed and synthesized in VHDL. The simulation was done by using Mentor Graphics Modelsim tool and synthesised using Quartus II version 12.1. The design is targeted for FPGA Cyclone IV GX EP4CGX150DF31I7AD technology. The FPGA implementation offers better speed and effective and optimised hardware. FPGA device utilisation also shows minimum device utilisation. The controller is able to integrate to a running cluster as well as initialize the communication as a leading cold start node. It contains configurable number of RX and TX registers, it supports many options according to the standard and it even contains options which are not allowed in the FlexRay standard. These options allow testing a wide range of parameters of the FlexRay node or cluster.

V. REFERENCES

[1] T. Fuhrer,B.Muller, W. Dieterle ,F. Hartwich,R. Hugel and M. Walther "Time triggered communication on CAN", Robert Bosch GmbH, In proc of 7th International CAN Conference, 2000.

[2] J. Ferreira, P. Pedreiras, L. Almeida and J. A.Fonseca, "The FTT-CAN protocol for flexibility in safety-criticalsystems" IEEE Micro, pp. 0272-1732, July–August 2002

[3] J. Pimentel, J. Kaniarz, "A CAN-based application level error detection and fault containment protocol"A Fault Management Protocol for TTP/C. In Proc of IECON'01 (Int. Conf. On Industrial Electronics), pp. 1800- 1805, Denver,2004

[4] F.Hartwich, B.Müller,T. Führer and R. Hugel, "CAN Network with Time Triggered Communication", Robert Bosch GmbH ,In proc of International CAN Conference, 2005.

[5] J. J. Nielsen and H. P. Schwefel, "Markov chain-based Performance evaluation of FlexRay dynamic segment," in Proc of Int. Workshop Real Time Netw., 2007, pp. 1–6.

[6] S.C. Talbot and S. Ren, "Comparison of FieldBus Systems, CAN, TTCAN, FlexRay and LIN in Passenger Vehicles" IEEE Int. Conf. on Distributed Computing Syst., pp. 1545-0678, 2009

[7] M. Lukasiewycz, M. Glab, J. Teich, and P. Milbredt, "FlexRay schedule optimization of the static segment," in Proc. Int. Conf. Hardware/Software Codes. Syst. Synthesis (CODES+ISSS), 2009, pp. 363–372

[8] E. G. Schmidt and K. Schmidt, "Message scheduling for the FlexRay protocol: The dynamic segment," IEEE Trans. Veh. Technol., vol. 58, no. 5, pp. 2160–2169, Jun. 2009.

[9] A.Forsberg and J.Hedberg,"Comparison of FlexRay and CAN-bus for real-time communication" Malardalen University, Hogskoleplan 1, afg05001@student.mdh.se, 2011

[10] S. R. Gadekar and S. Kodgire, "LIN protocol-emerging trend in automotive electronics", International Journal of Engineering Research and Technology. ISSN 0974-3154 Volume 6, pp. 507-514, nov 2013

[11] A. U. Jadhav and N.M. Wagdarikar, " A Review: Control Area Network (CAN) based Intelligent vehicle system for driver assistance using Advanced RISC Machines (ARM)",IEEE Int. Conf. on Pervasive Computing (ICPC), pp.978-1-4799-6272-3, 2015

[12] S. Shreejith and S. A. Fahmy, "Extensible FlexRay communication controller for FPGA-based automotive systems", IEEE transactions on vehicular technology, vol. 64, no. 2, february 2015

[13] F. Consortium, "FlexRay Communication System, Protocol Specification, Version 2.1 Rev.A," FlexRay Consortium, 12 December 2005.