Design of Flexray Protocol with high speed and area optimized for Automobile with modified FSM controller

[1]Priya Pararha, [2]Dr. Vinod Kapse

[1]M. Tech. Student, [2]Professor
[1,2]GGITS, Jabalpur

***Abstract*: Modern automobiles are not merely mechanical devices, but they are becoming highly sophisticated by including more and more functions which are controlled by small digital computers known as Electronic Control Units (ECUs). As new functions are included, there is not only demand on ECU, but there is an increasing demand on Communication networks placed in the automobile. FlexRay is a scalable, flexible, high-speed, deterministic, error tolerant communication technology that is designed to meet growing safety related challenges in the automobile industry. Synchronization between different nodes in a communication network is very essential for the proper working of a system. The clock synchronization service in FlexRay protocol suggests a distributed control system. In this thesis the clock synchronization algorithm named Fault Tolerant Midpoint (FTM) Algorithm is studied and its implementation in Normal Active State of Protocol Operation Control (POC) Module is achieved with the help of Vertex IV FPGA.**

***Keywords*: POC-Protocol Operation Control, CHI-controller host interface, CRC-Cyclic Redundancy Code, CSP-Clock Synchronization Process**

I-INTRODUCTION

The FlexRay protocol mechanisms described in this specification are presented using a graphical method loosely based on the Specification and Description Language (SDL) technique described in [10]. The intent of this description is not to provide a complete executable SDL model of the protocol mechanisms, but rather to present a reasonably unambiguous description of the mechanisms and their interactions. This description is intended to be read by humans, not by machines, and in many cases the description is optimized for understandability rather than exact syntactic correctness. The SDL descriptions in this specification are behavioral descriptions, not requirements on a particular method of implementation. In many cases the method of description was chosen for ease of understanding rather than efficiency of implementation. An actual implementation should have the same behavior as the SDL description, but it need not have the same underlying structure or mechanisms. Several SDL diagrams have textual descriptions intended to assist the reader in understanding the behavior depicted in the SDL diagrams. Some technical details are intentionally omitted from these explanations. Unless specifically mentioned, the behavior depicted in the SDL diagrams takes precedence over any textual description.

Author	Work	Outcome
Vinay B. Bhasale	superior flexibility, reliability and a high-speed communications Flexray protocol	63 slices of Vertex FPGA with Max operating frequency of 234 Mhz
P.Satya Shree Sai Ram	FPGA-based communication controller Time-triggered protocols FlexRay with event-triggered medium access used	65 slices of Vertex FPGA
Milind Khanapurkar	VHDL and FPGA Implementation of Communication Controller of FlexRay Controller	84 slices of Vertex FPGA
Naveena S George	FPGA implementation of Flexray clock synchronization module in normal active state of POC module	73 slices of Vertex FPGA

Table 1 Literature Summary

The Modern top of the line vehicles incorporate hundred or more embedded processing units which realizes advanced capabilities like pedestrian detection with auto-park, auto-brake and other comfort or safety features. These algorithms execute complex processing on information gathered from a network of sensors, to deliver control sequences for disseminated actuators. The data bandwidth and quality of service necessary for such advanced ECUs (electronic control units) exceeds the capabilities of the event-triggered Controller Area Network (CAN) protocol that has been used in automotive systems until now [1]. Moreover, new generation in-vehicle systems, specifically in electric cars that have level of automation and claims higher determinism, Available conventional CAN protocol is not much suitable for this.

II-IMPLEMENTATION

Proposed Communication controller (CC) is responsible for implementing the protocol aspects of the Flex Ray communication system. It performs all communications tasks such as reception and transmission of messages in a TTP cluster without interaction of the host CPU. Communication controller (CCThe FlexRay communication controller executes the communication protocol defined in the FlexRay specification. The primary tasks of the FlexRay communication controller include framing, bus access, error

detection and handling, synchronization, putting the FlexRay bus to sleep and waking it up, as well as coding messages which has to be transmitted and decoding messages which are received. The FlexRay communication controller in a FlexRay node consists of six modules as shown in the figure 3.2. They are controller host interface (CHI), protocol operation control (POC), coding and decoding (CODEC), media access control (MAC), frame and symbol processing (FSP), and clock synchronization process (CSP).

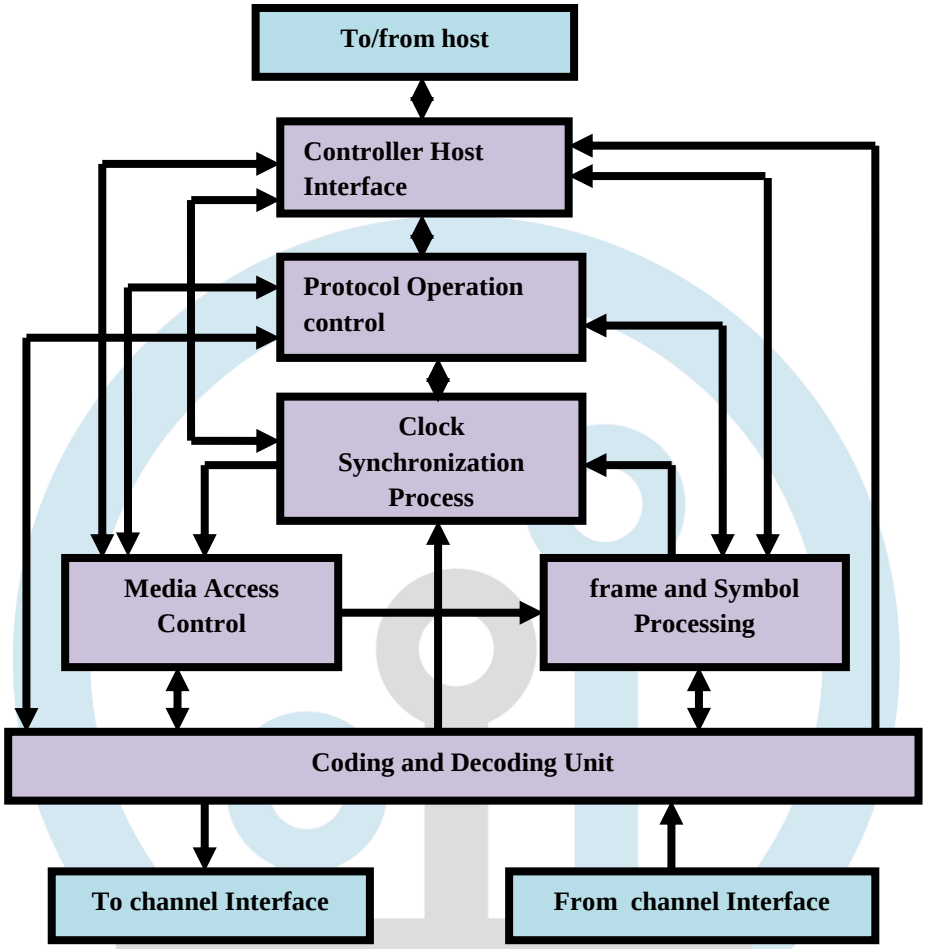

Fig 1 Proposed FlexRay Communication Controller Architecture

The CHI module acts as a mediator between the host controller and the FlexRay communication controller within each node. The POC module controls the operational modes of the FlexRay modules. The CODEC module is responsible for encoding the communication elements into the form of a bit stream and for receiving communication elements, making bit streams and checking the correctness of the bit streams. The MAC module controls the access of FlexRay node to the
bus. The FSP module is responsible for checking the correct timing of received frames and symbol, applying further syntactical tests to received frames, and checking the semantic correctness of received frames. The CSP module is responsible for generating timing units in the FlexRay communication controller, e.g., communication cycles. Moreover this module uses a distributed clock synchronization mechanism in which each node individually synchronizes itself to its cluster by observing the timing of transmitted frames from other nodes.

CLOCK SYNCHRONIZATION MODULE: Synchronization is the process of making different actions in a communication network with distributed clock system to agree on a same time reading. In FlexRay protocol each ECU's are having their own local clocks whose values deviates from each other as time passes due to problems like voltage and temperature variations, even though they are initially synchronized with each other.
The basic time unit generated by the crystal oscillator in each ECU is known as Microtick. The durations of microticks are not constant, and to solve this problem a single, unique concept of time which is simplified to Global time is required. The Global time is expressed in terms of Macroticks. A fixed number of Macroticks are then combined together to form the communication cycles. Possible deviations in a clock take place in rate and offset parameters. These deviations are detected and corrected with the help of an algorithm named as Fault Tolerant Midpoint (FTM) algorithm.

ACTIVE STATE IN POC MODULE: POC is responsible for changing the mode of the core mechanisms of the protocol in response to changing conditions in the node. Table 3.1 depicts an overview of the communication controller POC operations. Among the several states of POC module the Default config state is the startup state of communication controller. Config state sets

up all the necessary conditions for the protocol's proper functioning. Ready state indicates it is ready for communication. If the node is in sleep mode Wakeup state can be used to make it active. During Startup state it initiates the communication process The states of communication controller and respective commands are as in state table

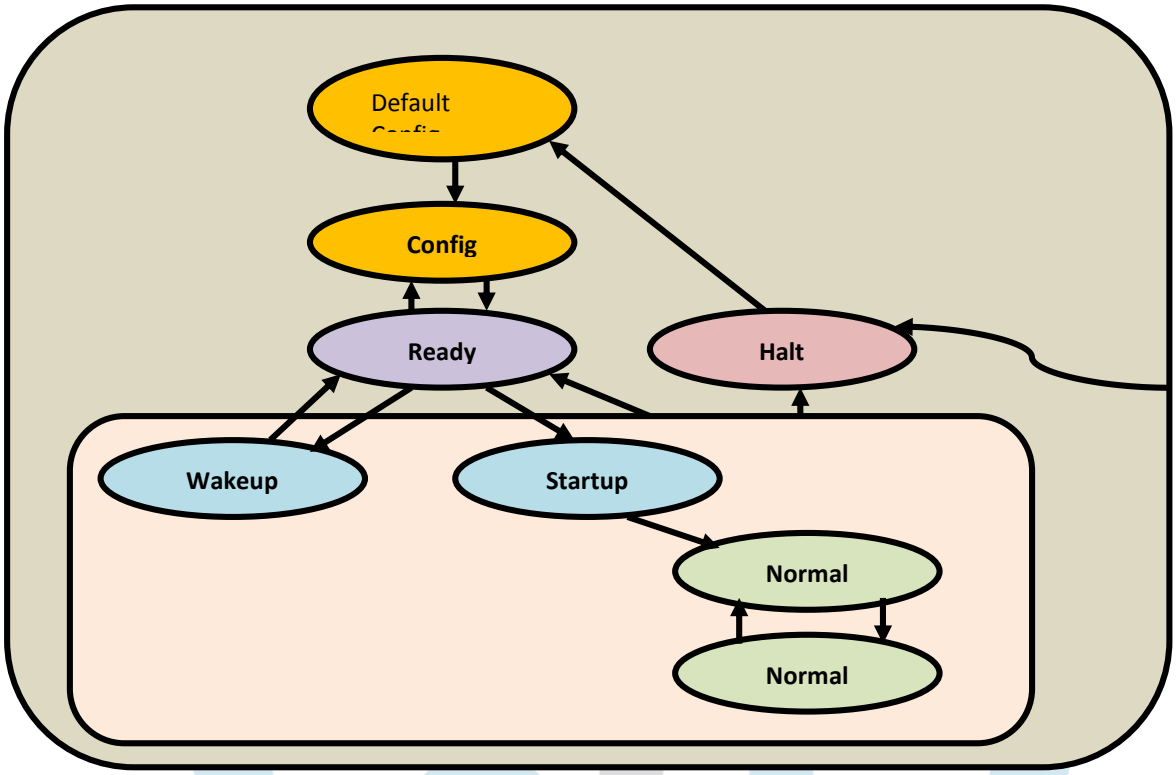

Fig 2 State diagram of FlexRay Protocol Operation Control

STATE_TYPE	COMMAND [4-0]
DEFAULT_CONFIG	00000
CONFIG	00001
READY	00010
HALT	00011
WAKEUP	00100, 00101, 00110
Startup	00111, 01000, 01001
NORMAL_ACTIVE	01010
NORMAL_PASSIVE	01011

Table 1. State Table.

In Normal Active state the POC performs Synchronization calculations at the end of each cycle to determine whether the POC should change the modeling of the core mechanisms before the beginning of the next communication cycle. The state changes occur in accordance with the sync calculation results received during this state. If the sync calculation results are within bound and there are no errors the POC will remain in the active state itself. Else it will transfer to Passive or HALT conditions according to the error.

DEFAULT CONFIG state: The CC enters this state when leaving hard reset or when exiting from HALT state. To leave DEFAULT CONFIG state, the Host has to write Command [4:0] = \00001"' then transits to CONFIG state.

CONFIG state: The CC enters this state when exiting from DEFAULT CONFIG state or when exiting from READY state. After unlocking CONFIG state and writing command [4:0] = \00010"' the CC enters READY state. From this state the CC can transit to WAKEUP state and perform a cluster wakeup or to STARTUP state to perform a cold start or to integrate into a running cluster. The CC enters this state

Ready State: When exiting from CONFIG,WAKEUP, STARTUP, NORMAL ACTIVE, or NORMAL PASSIVE state by writing command [4:0] = \00010"' (READY command). The CC exits from this state To CONFIG state by writing command [4:0] = \00001" (CONFIG command) and To WAKEUP state by writing command [4:0] = "00100" (WAKEUP command) and To STARTUP state by writing command [4:0] = "00111" (STARTUP command).

WAKEUP State: CC enters this state when exiting from READY state by writing command 00100" (WAKEUP command). The CC exits from this state to READY state after complete non-aborted transmission of wakeup pattern or after WUP reception or after detecting a WUP collision or after reception of a frame header or by writing command [4:0] = \00010" (READY command)

STARTUP State: Any node entering STARTUP state that has cold start capability should assure that both channels attached have been awakened before initiating cold start.
NORMAL ACTIVE state: Cold start path initiating the schedule synchronization or Cold start path joining other cold start nodes or Integration path integrating into an existing communication schedule (all other nodes). A cold start attempt begins with the transmission of a collision avoidance symbol (CAS).

III-DESIGN APPROACH

A cluster consisting of three FlexRay nodes is considered for this paper. Each node is assumed to have a local clock and the frequencies of these clocks are 80 MHz, 100 MHz and 60 MHz respectively. The Macrotick deviation in each clock after some time is shown in Table 2.

Parameters	Values	Units
Macrotick duration (Required)	1	MHz
Duration of Macrotick generated by 80 MHz clock after deviation	844.3	KHz
Duration of Macrotick generated by 100 MHz clock after deviation	1.14	MHz
Duration of Macrotick generated by 60 MHz clock after deviation	762.13	KHz

Table 2 MACROTICK DURATION AFTER DEVIATION

The values of rate deviations are calculated by comparing the microtick counter values of the clocks before and after the possible deviations. The values of offset deviations are calculated by using a phase frequency detector. A phase frequency detector is a logic circuit that generates a voltage signal which represents the difference in phase between two signal inputs. After obtaining the phase and rate deviations they are arranged in descending order to perform the FTM algorithm. Since, the cluster under consideration contains only three nodes the value of k will be 1. Therefore one extreme value from the greatest and smallest measured values of the list is removed. Then the remaining one value is taken as the final offset and rate correction value. As per this value, that much number of Microticks are added or subtracted from each deviated Macrotick to make them synchronous with other Macroticks. The hardware description language used for developing the design was VHSIC Hardware Description Language (VHDL). Mentor Graphic's Modelsim was used for simulating the VHDL design.

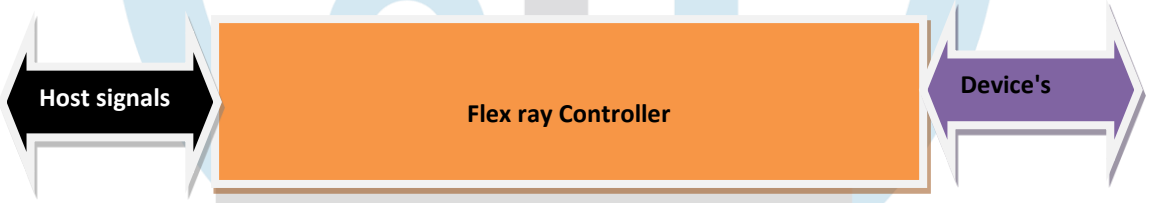

Figure 3: FLEXRAY PROTOCOL Controller conversion & vice versa

Figure 3 shown below is our presented design, we have presented to design an FLEXRAY PROTOCOL, it's been design using Pipelining & Mealy FSM been used these modification gives us Area & speed optimized Flex ray Controller
The presented 16-Slot Controller provides:-
An interface in-between high-speed FLEXRAY PROTOCOL domain & low-power domain.
The Controller appears as a slave on FLEXRAY PROTOCOL, whereas on , it is master.
Read & write transfers on FLEXRAY PROTOCOL are converted into corresponding transfers on. As it is pipelined, wait states are added during transfers to & from when FLEXRAY PROTOCOL is required to wait for protocol.
The FLEXRAY PROTOCOL to Controller comprises of a state machine, which is used to control generation of & FLEXRAY PROTOCOL output signals, & address decoding logic which is used to generate peripheral select lines.
All registers used in system are clocked from rising edge of system clock HCLK, & use asynchronous reset HRESETn.

III-RESULTS

We are optimizing area & seed both that's why design goal is balanced.
Timing Summary:
Timing Summary: Speed Grade: -11
Minimum period: 2.589ns (MFlexray Protocolmum Frequency: 316.250MHz)
Minimum input arrival time before clock: 3.927ns
MFlexray Protocolmum output required time after clock: 5.057ns
MFlexray Protocolmum combinational path delay: 5.946ns
Total 5.946ns (4.634ns logic, 1.312ns route)
(77.9% logic, 22.1% route)

Table's below shows synthesis results of presented work.
Device: xc4vlx200-11ff1513 (Vertex -4 FPGA)

axi2apb Project Status (09/06/2018 - 01:56:54)			
Project File:	Flexray.xise	**Parser Errors:**	No Errors
Module Name:	Flexray	**Implementation State:**	Synthesized
Target Device:	xc4vlx25-12ff676	**•Errors:**	No Errors
Product Version:	ISE 12.2	**•Warnings:**	10 Warnings (10 new)
Design Goal:	Balanced	**•Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**•Timing Constraints:**	
Environment:	System Settings	**•Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	57	10752	0%
Number of Slice Flip Flops	49	21504	0%
Number of 4 input LUTs	111	21504	0%
Number of bonded IOBs	238	448	53%
Number of GCLKs	1	32	3%

Figure 4 Synthesis Result

Number of Slices	**57**	**out of**	**89088**	**0%**
Number of 4 input LUTs	111	out of	178176	0%
Number of bonded IOBs	238	out of	960	24%
IOB Flip Flops	49	out of	178176	0%
Number of GCLKs	1	out of	32	3%

Table 3 synthesis results obtain

Figure 5 below shows logical representation of presented design. In RTL result it may be clearly monitor all hierarchy maintained also FLEXRAY PROTOCOL signal & signals at top level of abstraction. All interconnect of sub-modules in RTL are connected it shows correct abstraction & signal flow of presented work. An extended RTL design of presented work is shown in figure 5 it has all connections & all internal module available means correct coding & design is synthesis properly no any open element found in design.

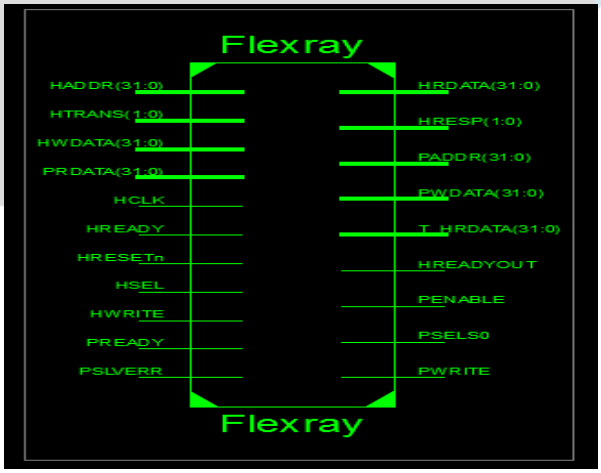

Figure 5 RTL top view

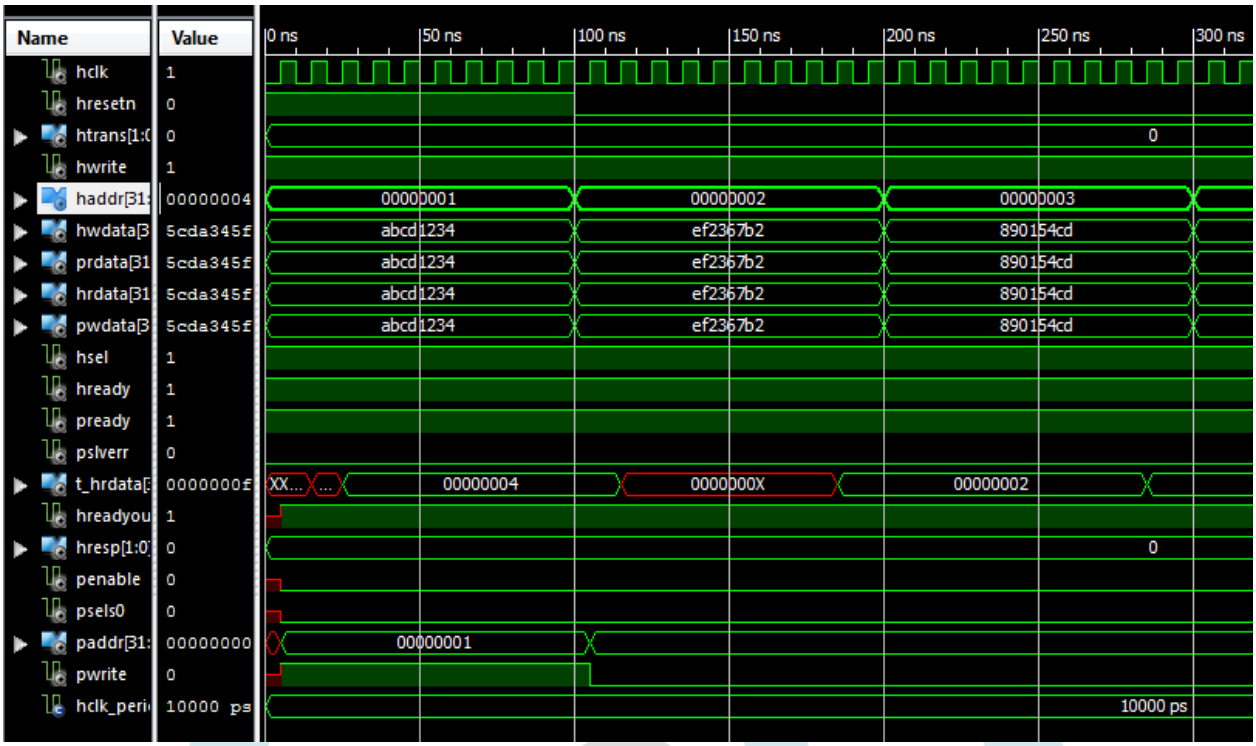

Figure 6 Simulation of FLEXRAY PROTOCOL

In Simulation 'Hwdata' is same as 'Prdata' & 'Hrdata' is same as 'Pwdata' for all various address, it shows correct simulation.

	Naveena S George	**Milind Khanapurkar**	**P.Satya Shree Sai Ram**	**Vinay B. Bhasale**	**Proposed**
Slices	73	84	64	63	57
Max freq.	-	-	-	234 Mhz	316.250 MHz

Table 5 Comparative results

Form table above above its may be monitor that number of slices in presented design for vertex FPGA is less as compare with base work by Vinay B. Bhasaleet al [1] & P.Satya et al [2], As in vertex FPGA 1 slice is equals to 2 LUT & 1 flip flop and, LUT of Vertex FPGA has 4 input & 1 output PLA, hence in presented work for 57 slice total 114 LUT's or 4x1 PLA used & total 57 flip flop used. In Vinay B. Bhasale works total 132 slices used means 264 LUT's & 132 flip flops used. In P.Satya works total 77 slices used means 154 LUT's & 77 flip flops used. extreme frequency obtain in presented design is high as compare with available work.

IV-CONCLUSION

This thesis has highlighted the concept of the FlexRay protocol. Authors have designed the communication Controller of ex ray node with FSM and the simulation and synthesis results on xilinx tool are presented. The presented work demonstrated that verification of automotive software is feasible. However, feasibility alone is not sufficient for the method to become accepted by car manufacturers. It is also the question of the development cost that is vital. The VHDL model Bus guardian module of ex ray node is already designed and verified, in future authors have planned to integrate communication controller and Bus Guardian with suitable host and use this integrated assembly for some intra vehicular communication application.

REFERENCES

[1] Vinay B. Bhasale, Nitesh Guinde, Modelling and Analysis of Flexray Protocol in VHDL, Proceedings of the IEEE 2017 International Conference on Computing Methodologies and Communication (ICCMC), 978-1-5090-4890-8/17/2017 IEEE

[2] P.Satya Shree Sai Ram, Mr.S.Yuvaraj, FLEXRAY COMMUNICATION CONTROLLER FOR FPGA-BASED AUTOMOTIVE SYSTEMS AS AN IP CORE, International Journal of Advances in Engineering Research http://www.ijaer.com (IJAER) 2015, Vol. No. 9, Issue No. V, May ISSN: 2231-5152

[3] Milind Khanapurkar1, Jayant Y. Hande2 and Dr. Preeti Bajaj, Approach for VHDL and FPGA Implementation of Communication Controller of FlexRay Controller, Journal of Information Hiding and Multimedia Signal Processing c 2010 ISSN 2073-4212 Ubiquitous International Volume 1, Number 4, October 2010

[4] Jirí Záhora, Martin Vajnar Automotive control unit with CAN and FlexRay, Faculty of Electrical Engineering, Master's Thesis, Czech Technical University in Prague

[5] Naveena S George, Shan Mary Cherian Sherin Mary Enosh Jiss Paul, FPGA IMPLEMENTATION OF FLEXRAY CLOCK SYNCHRONIZATION MODULE IN NORMAL ACTIVE STATE OF POC MODULE, International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2163 Volume 1 Issue 9 (October 2014) www.ijirae.com

[6] Next Generation Car Network - FlexRay, Fujitsu Microelectronics (Shanghai) Co.,Ltd. Jun 2006

[7] Awani Gaidhane, Milind Khanapurkar, and Preeti Bajaj , Design approach for FPGA implementation of ex-ray controller using VHDL for intra vehicular communication application, Proc. of International Conference ICETET, G. H. Raisoni College of Engineering, Nagpur, 2008.
[8] P. M. Szecowka, and M. A. Swiderski, On hardware implementation of FlexRay bus guardian module, Proc. of the 12th MIXDES. FlexRay Communication System-Protocol Speci_cation, v2.0, FlexRay Consortium, 2007.
[9] FlexRay Consortium, Homepage, http://www.exray-group.org
[10] FlexRay Consortium, FlexRay Communications System: Protocol Speci_cation Version 2.0, 2004.