OPEN ACCESS

International Journal Of Modern Engineering Research (IJMER)

Design Of FPGA Based Flexray Communication Protocol For Superior Automotive Applications

*S.Sedhumadhavan[1], M.Tech ,M.Anitha[2], B.Renukadevi[2], B.Vinothini[2]

[1]*Assistant Professor, Department Of ECE, Rajiv Gandhi College Of Engineering & Technology, Puducherry, India,*

[2]*UG , Department Of ECE, Rajiv Gandhi College Of Engineering & Technology, Pondicherry University, India,*

*Corresponding Author: *S.Sedhumadhavan*

ABSTRACT: *Recent Vehicle integrates many numeral of Electronic components were increased rapidly during recent years. The automotive embedded systems have great need for dependability, on designing the FlexRay protocol. FlexRay (FR) protocol is mainly for scalable, flexible, high speed deterministic, error tolerant communication in order to meet growing safety related challenges in the automobile industry. FlexRay uses time triggered protocol for gaining the high speed sterling communication marking a fluctuation from the even triggered medium access used in CAN. FlexRay enables the impeachability proletarian for next generation shatterproof application. This paper explores the general issues of functional coverage pertaining to the FlexRay specification. This paper describes an FPGA based communication controller. It is rigged out and substantiated on a Xilinx Spartan 6 FPGA. It is combined with both a logic based hardware ECU and a fully fledged processor based ECU. Result shows that rigged out generate a highly systematic core in terms of power, performance and resource. The proposed system has many advantages like timeliness, fault tolerance and security.*

Keywords: *Electronic Control Unit (ECU), Field Programmable Gate Array (FPGA), Communication Controller (CC), Time Division Multipexing Access (TDMA, Controller Area Network (CAN), Xilinx*

I. INTRODUCTION

Modern high-end vehicles incorporate one hundred or more embedded reckonings units which rigged out advanced potentiality like auto-park, pedestrian detection with auto-brake and other safety or comfort features. Nowadays, automotive systems are complex distributed embedded system to compose several Electronic Control Unit 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 with a very high level of security but it doesn't met the set of challenging conditions & requirements. In general, CAN is based on an event-driven communication approach because each bus node of a communication system must able to access common Communication medium with data rate of 1 Mbit/sec. For high data rate, the CAN bus didn't 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.

Remainder of this paper is organized as follows. In section II describes the literature and related work in this area. In section III presents the proposed architecture of the FlexRay ECU and FlexRay Frame. In section IV gives brief introduction to communication controller. Section V presents simulation and synthesis results. Finally it concludes the paper and outline about the future work in section VI.

II. RELATED WORK

According to the criteria's communication protocols are classified into two types such as Event triggered and Time triggered protocol. Most prevailing bus system for in-vehicle communication use event triggered protocol that is CAN bus is at the bit rate of 1Mbit/sec. The Time triggered protocol TTP/C, message are transmitted at pre-defined time instants [5]. A extent higher bandwidth with 10Mbit/sec is provided by the

FlexRay bus. Moreover the FR protocol allows a time triggered communication for several control function with real time requirements [4].

CAN was developed in the 1980's to account for the perceived deficiencies of the I2C and D2B networks as used in automobiles at the time CAN has been extended to a mass of embedded electronic application domains, including domestic appliances, military equipment and medical devices. As an increasing number of embedded electronic components are added to vehicles in order to meet rising consumer demand for product features, the complexity of each embedded component as well as the complexity of groups or sub-systems of components is rising exponentially along with the onset of time. CAN is an example of a complex Cyber Physical System (CPS), or a "deeply" embedded electronic system with locally autonomous or semi-autonomous sensors and actuators, having components which are coordinated across a network.CAN uses short messages – the maximum utility load is 94 bits.. CAN is a serial field bus communication network [9].

Frame Format of CAN:
SOF (1 bit)- Start of Frame. The message starts from this point.
Identifier (11 or 29 bits) - The ID can contain source and destination addresses. The value of the Identifier determines the message priority. The lower the value, the higher the priority.
RTR (1 bit) – Remote Transmission Request.

- RTR 1- this packet is for destination address.
- RTR 0- this packet is request for data from device.

IDE (6 bits) – Single Identification Extension..

- 1st bit –identifier extension.
- 2nd bit- always 1.
- Last 4 bit-code for data length.

R0– reserved bit.
Data (0 to 8 bits) – Up to 64 bit of data can be transmitted. Length depends on data length code.
CRC (15 bits) – Cyclic Redundancy Check. It contains the checksum (number of bits transmitted) of the preceding application data for error detection.1 bit is for delimiter.
ACK (2 bit)– 1 bit is for ack and 2 bit for ack delimiter.
EOF (>=3 bits) – end of frame. It marks end of can frame and disables bit stuffing and ends with seven zeros.
IFS– Inter Frame Space. It contains the time required by the controller to move a correctly received frame to its proper position[8].

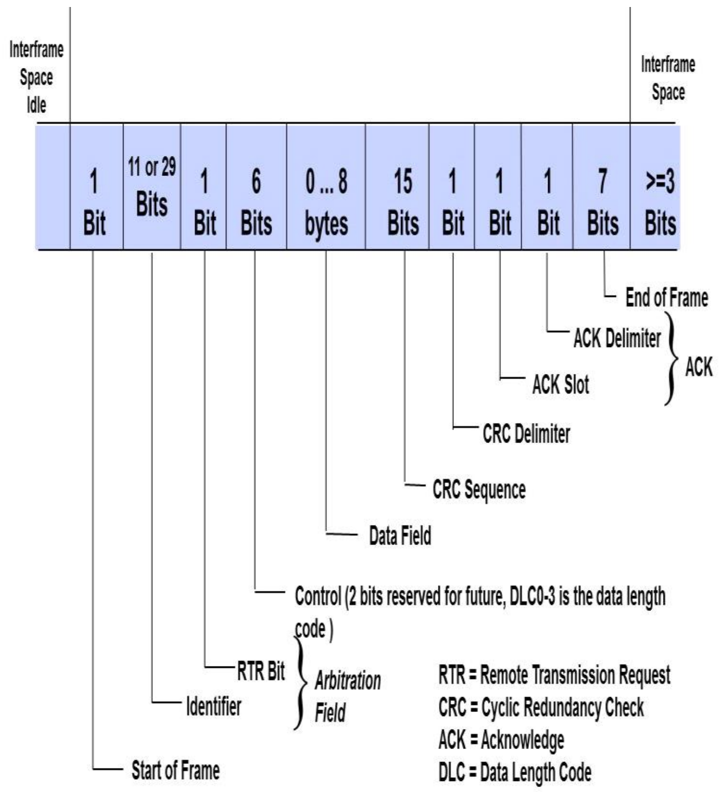


Figure1: Frame format of CAN

III. PROPOSED FLEXRAY

Flex ray protocol is a communication protocol. Developed by flex ray consortium in 1999 [6].It is used for Fault tolerant and flexibility. Flex ray system consists of ECU each with a bus interface connected to one or more communication channel. Such system is called FLEXRAY CLUSTER. The FlexRay communications bus is a deterministic, fault-tolerant and high-speed bus system developed in conjunction with automobile manufacturers and leading suppliers. FlexRay delivers the error tolerance and time-determinism performance requirements for x-by-wire applications (i.e. drive-by-wire, steer-by-wire, brake-by-wire, etc.). The FlexRay protocol is a unique time-triggered protocol [1].

.

A.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 [3].

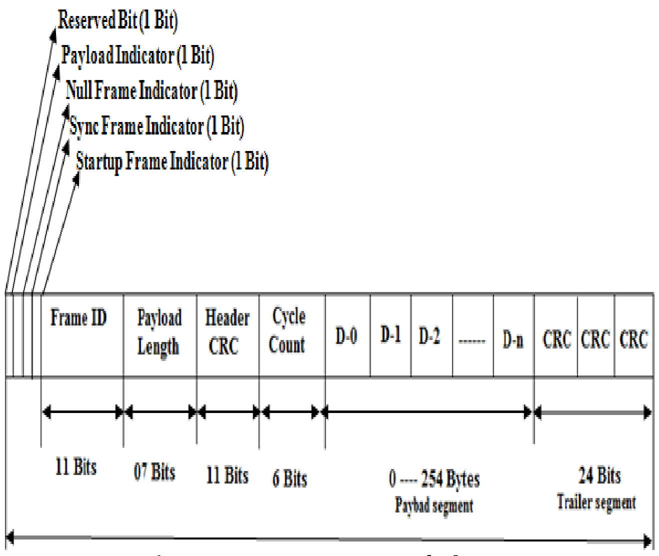


Figure 2: Frame Format of Flexray

Header:

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

- **Reserved bit (1 bit) -** It is reserved for future protocol use.
- **Payload preamble indicator (1 bit) –** Indicates the extension of vector information in the frames payload segment **Null frame indicator (1 bit) –** It is mainly to identify the frame is an empty frame. ”0” means the payload segment contains no valid data; “1” means the payload segment contains valid data.
- **Sync frame indicator (1 bit) –** It is to check the frame is a sync frame. “0” means no synchronization for node; “1” means all receiving nodes shall use the frame for synchronization if it meets synchronization conditions.
- **Startup frame indicator (1 bit) -** This bit indicates whether or not a frame is a startup frame. Only cold start nodes1 are allowed to transmit startup frame. “0” means this frame is not a startup frame; “1” means this frame is a startup frame. This part set to “1” in the sync frames of cold start nodes. A cold-start node can only transmit one frame per CC with startup frame indicator set to “1”.
- **Frame ID (11 bits) -** This position defines the slot in which the frame should be transmitted. Each slot has a slot number. If the slot number equals to frame ID, this slot can use for transmission of this frame. **Payload length (7 bits) -** This part is used to indicate the size of the payload segment. **Header CRC (11 bits) -** This part contains a cyclic redundancy check code (CRC) that is computed over the sync frame indicator, the startup frame indicator, the frame ID, and the payload length.
- **Cycle count (6 bits) -** This part indicates the value of cycle counter, from the transmitting node's view, at the time of frame transmission happened [6].

Payload:

The FlexRay payload segment contains 0 to 254 bytes data (0 to 127 two-byte words).

Trailer:

The FlexRay trailer segment is a 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].

A. FlexRay Communication Cycle

A FlexRay communication cycle consists of Static segment (ST) and Dynamic segment (DYN) as shown in Figure 3. Static slot uses the TDMA method and Dynamic slot use FTDMA (Flexible TDMA) for bus access [3].

(1) Static segments:

The ST segment uses static time-trigger, namely Time Division Multiple Access (TDMA), as the media access control. TDMA is a channel access method for shared medium networks. Each slot has a fixed length and identification (ID) which is assigned to a specific control unit at development time [7].

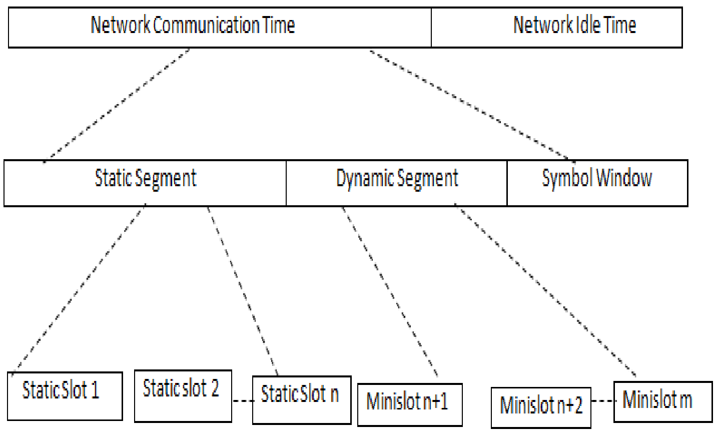


Figure 3: FR Communication Cycle

(2) Dynamic segments:

The DYN segment uses flexible time-trigger, namely Flexible Time Division Multiple Access (FTDMA), as the media access control. If task is triggered by a significant change of state, it needs to use FTDMA to arbitrate the media and be transmitted. FTDMA enables frames have chances to be sent whenever they require. There are no static slot allocations in advance. The media access is priority based [2].

(3) Symbol segment:

In the symbol segment, FlexRay sends internal control information. Moreover, specific bit sequences are sending at the boot process to wake up all nodes.

(4) NIT and Frames:

The Network idle time after the dynamic segment is used for the synchronization of the clocks. Each slot in the segments corresponds to one frame. A frame contains up to 254 bytes of data. Each controller only detects on the basis of the slot ID and its application layer whether the information is designated to it [1].

IV. FLEXRAY COMMUNIVATION CONTROL

The Communication Controller (CC) is a part of FlexRay ECU shown in Figure 4 and it is used for implementing the protocols aspects of FlexRay communication system. It performs all task of communication process such as reception and transmission of messages in time triggered protocols cluster without interaction of the host CPU.

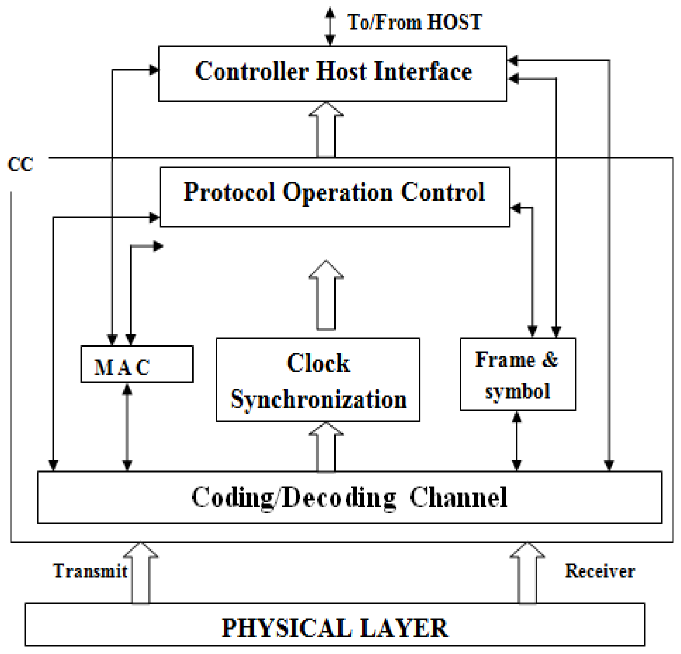


Figure 4: Block diagram of Communication Controller

A.**Communication controller states**

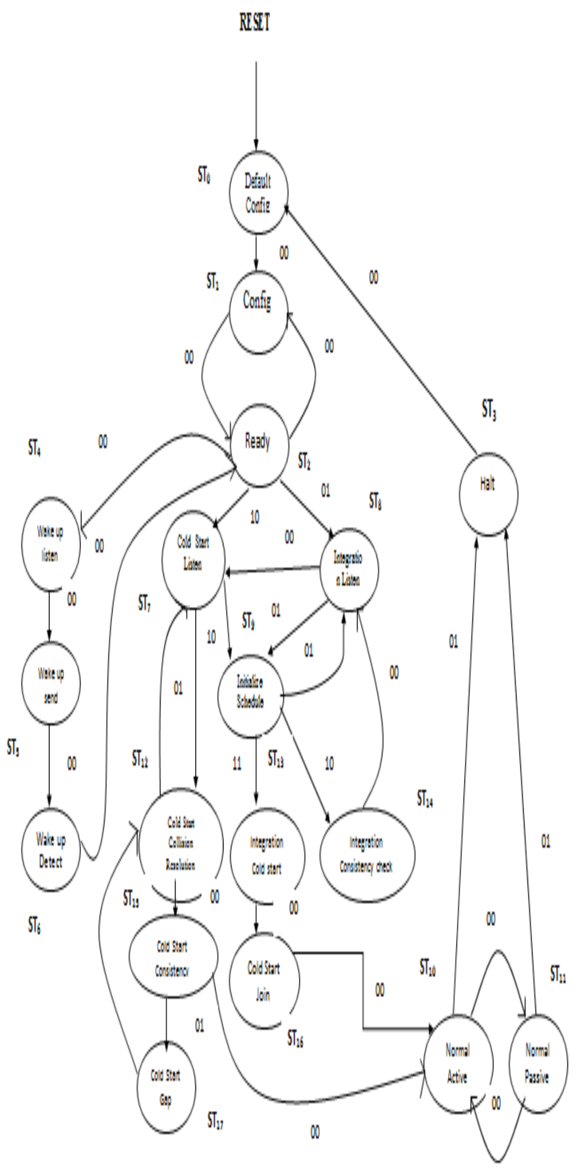
The states of communication controller and respective commands are as in state table Figure 5 and the state flow is as described in state diagram shown in Figure 6.

State Type	Command [4-0]
Default_Config	00000
Config	00001
Ready	00010
Halt	00011
Wakeup_Listen	00100
Wakeup_Send	00101
Wakeup_Detect	00110
Coldstart_Listen	00111
Integration_Listen	01000
Initialize_Schedule	01001
Normal_Active	01010
Normal_Passive	01011
Coldstart_Collision_Resoultion	01100
Intergration_Coldstart_Check	01101
Intergration_Consistency_Check	01110
Coldstart_Consistency_Check	01111
Coldstart_Join	10000
Coldstart_Gap	10001

Figure 5: State Table

- **Default Config state**: The CC enters this state when it 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 only when it's exiting from DEFAULT CONFIG state or when it's from READY state. After unlocking CONFIG state and writing command [4:0] = "00010"' the CC enters READY state. From this state the CC 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.
- **Ready State**: When exiting from CONFIG, WAKEUP, STARTUP, NORMAL ACTIVE, or NORMAL PASSIVE state by writing command [4:0] = "00010"'
- **Wakeup State**: CC enters this state when exiting from READY state by writing command [4:0] = "00100"

- The Wakeup Listen state is controlled by the wakeup timer and the wakeup noise timer. The two timers are controlled by the parameters listen timeout and listen timeout noise. Listen timeout enables a fast cluster wakeup in case of a noise free environment, while listen timeout noise enables wakeup under more difficult conditions regarding noise interference. In WAKEUP SEND state the CC transmits the wakeup pattern on the configured channel and checks for collisions. After return from wakeup the Host has to bring the CC into STARTUP state by CHI command RUN. In WAKEUP DETECT state the CC attempts to identify the reason for the wakeup collision detected in WAKEUP SEND state.
- **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) [8].



V. SIMULATION AND SYNTHESIS RESULT

The communication controller is designed using FSM methodology. The state coding and command sequence are as per the table, Frame structure, protocol operation using Verilog code and Xilinx tool (version 14.1) are shown below. In Figures 6, 7 & 8 shows the design of proposed FSM states and its RTL Viewer

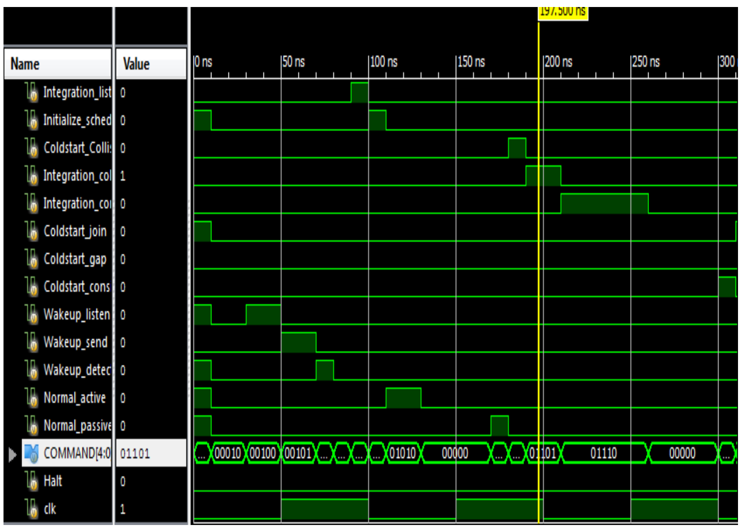


Figure.6 Design of proposed FSM states

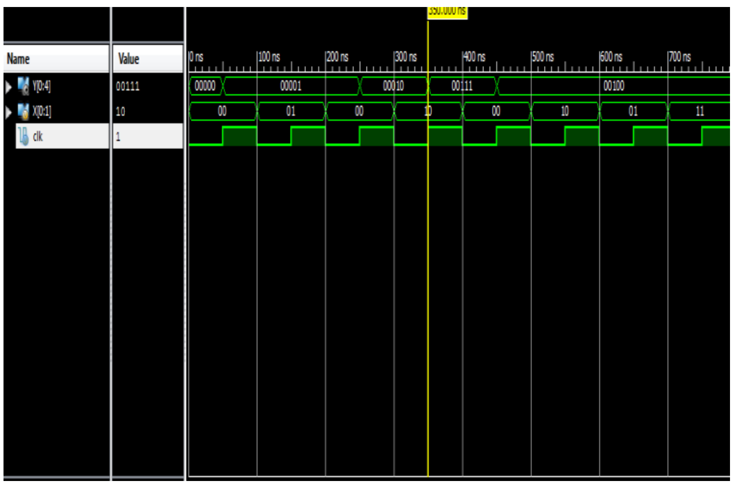


Figure.7 Design of Protocol operation

- **Frame Structure:** The output of frame depends on input command (4-0), clock 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"). Figures 9 & 10 shows the proposed frame structure and its RTL viewer.
-

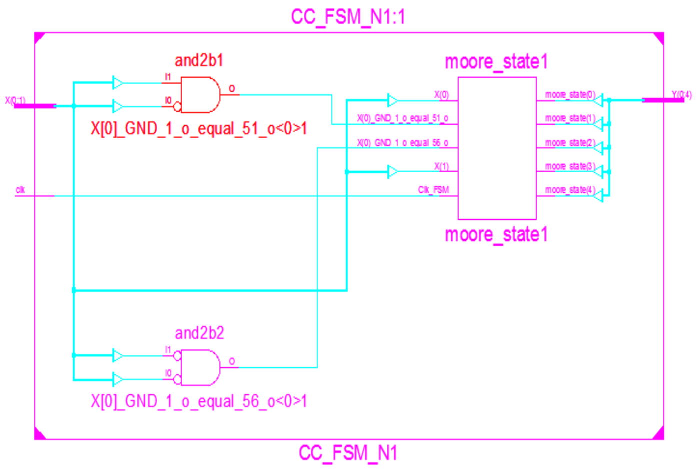


Figure.8 RTL Viewer of proposed FSM states

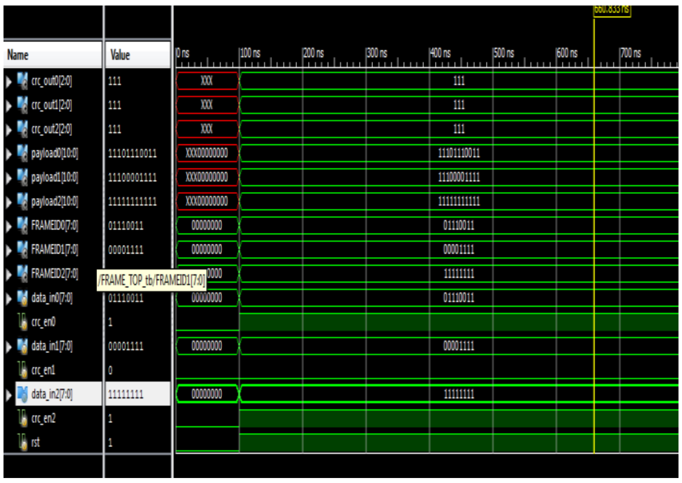


Figure.10 Design of proposed Frame structure

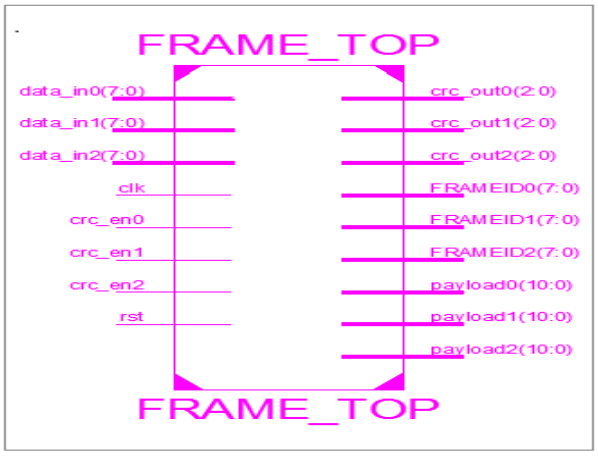


Figure.9 RTL Viewer of proposed Frame structure

VI. CONCLUSION AND FUTURE SCOPE

This paper highlighted the concept of FlexRay protocol and its functional specification. The presented work demonstrated the design of Communication controller of FlexRay node and simulation & synthesis report are shown using Xilinx. Finally the principle demonstrated in this paper are applicable for other time triggered interface and we hope to explore this for time triggered Ethernet

REFERENCES

[1]. Shanker Shreejith Student Member, IEEE and Suhaib A. Fahmy Senior Member "Extensible FlexRay Communication Controller for FPGA-Based Automotive Systems" IEEE Transaction on Vehicular technology, issue 99, May 2014

[2]. S. Shreejith, K. Vipin, S. A. Fahmy, and M. Lukasiewycz, "An Approach for Redundancy in FlexRay Networks Using FPGA Partial Reconfiguration," in Proceeding of Design, Automation and Test in Europe (DATE) Conference, pp. 721–724, Mar2013

[3]. J. Sobotka and J. Novak, "FlexRay controller with special testing capabilities," in Proc. International Conference on Applied Electronics (AE), pp. 269–272Y, Sep2012

[4]. C. Schmutzler, A. Lakhtel, M. Simons, and J. Becker, "Increasing energy efficiency of automotive E/E-architectures with Intelligent Communication Controllers for FlexRay," in Proceeding of International Symposium on System on Chip (SoC), pp.92-95, Oct 2011

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

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

[7]. X. HeI, Q. Wang, and Z. Zhang, "A Survey of Study of FlexRay Systems for Automotive Net," in Proceeding of International Conference on Electronic and Mechanical Engineering and Information Technology, pp.1197-1204, Aug 2011

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

[9]. A. Hagiescu, U. Bordoloi, S. Chakraborty, P. Sampath, P. Ganesan, and S. Ramesh, "Performance analysis of FlexRay-based ECU networks," Design Automation Conference, pp. 284-289, June 2007

*S.Sedhumadhavan. "Design Of FPGA Based Flexray Communication Protocol For Superior Automotive Applications." International Journal Of Modern Engineering Research (IJMER), vol. 07, no. 10, 2017, pp. 85–93.