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

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

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

***Abstract*: The FlexRay communication system is an emerging standard for advanced automotive control applications, such as drive-by-wire. The detailed micro-architectural level of the FlexRay specification facilitates implementations where the internal operation closely set the protocol specification definitions. Corresponding functional coverage models in the verification environment can also be defined relative to the FlexRay specification which enables consistent understanding, easier maintenance and better reuse. This thesis explores the general issues of functional coverage pertaining to the FlexRay specifications. VHDL simulation & FPGA synthesis results of communication controller of Flexray controller are tested on xilinix presented.**

***Keywords*: BIST-Built-In Self-Test, POC-Protocol Operation Control, FTM-Fault-Tolerant Midpoint**

I-INTRODUCTION

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. Bit rates The FlexRay Communications System specifies three standard bit rates - 10 Mbit/s (corresponding to a nominal bit duration, gdBit, of 100 ns), 5 Mbit/s (corresponding to a gdBit of 200 ns), and 2.5 Mbit/s (corresponding to a gdBit of 400 ns). In order to be considered FlexRay conformant, a protocol implementation is required to support all three standard bit rates.

Synchronization methods: A FlexRay node supports three different synchronization modes. The behavior of the cluster depends on the employed synchronization mode of the nodes in the cluster. 1.10.1 TT-D synchronization method A cluster in which the coldstart nodes use the TT-D synchronization method is a TT-D cluster. The TT-D synchronization method uses a distributed algorithm to reduce the effect of any single failure. No critical task depends on any single node.

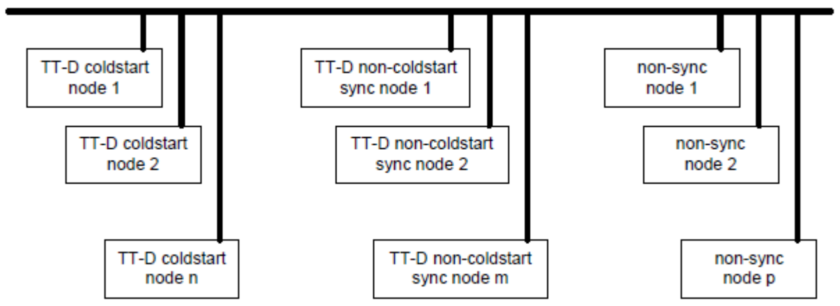

Figure 1: TT-D cluster.

Figure 1 shows the configuration of nodes in a TT-D cluster. The number of TT-D coldstart nodes n must be equal to or greater than 2 and the sum of the number of TT-D coldstart nodes n and the number of TTD non-coldstart sync nodes m must be equal to or less than 15. The number of non-sync nodes p is not limited by the protocol.

TT-L synchronization method: A cluster in which the sole coldstart node uses the TT-L synchronization method is a TT-L cluster. The TTL synchronization method is a modification of the TT-D synchronization method that reduces the number of required coldstart nodes from two to one. The single TT-L coldstart node in a TT-L cluster essentially behaves like two regular TT-D coldstart nodes by transmitting two startup frames. In this way non-sync nodes of the TT-L cluster will behave as if they were placed in a TT-D cluster with two TT-D coldstart nodes regularly transmitting their startup/sync frames and will integrate and operate normally, unaware of the fact that the two frames they receive actually come from the same node. The schedule and timing of such a TTL cluster will depend entirely on the single TT-L coldstart node.

The advantages of the TT-L synchronization method are a reduced system complexity, a slightly reduced startup time, and an improved precision. Figure 2 shows the configuration of nodes in a TT-L cluster. There exists exactly one TT-L coldstart node.The number of non-sync nodes p is not limited by the protocol.

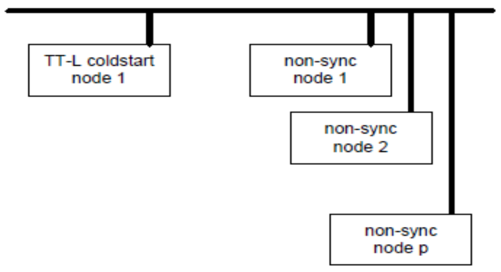

Figure 2: TT-L cluster.

TT-E synchronization method: A cluster in which the coldstart nodes use the TT-E synchronization method is a TT-E cluster. The primary intent of the TT-E synchronization method is to synchronize the schedule of the TT-E cluster, also called a time sink cluster, to a second FlexRay cluster, which is referred to as time source cluster. To that end, each TT-E coldstart node, also called a time gateway sink node, must be paired with a node of its time source cluster; this pair of nodes is called a time gateway. The node on the time sink cluster side is then called a time gateway sink node while the node on the time source cluster side is called time gateway source node. Figure 3 depicts the basic setup. Depending on the synchronization method employed by time source cluster, the time gateway source node may be a TT-D coldstart node, a TT-D noncoldstart sync node, a TTL coldstart node, or a non-sync node.6 The two nodes of the time gateway are connected via a time gateway interface, which is used by the time gateway source node to provide information about the schedule of the time source cluster to the time gateway sink node.

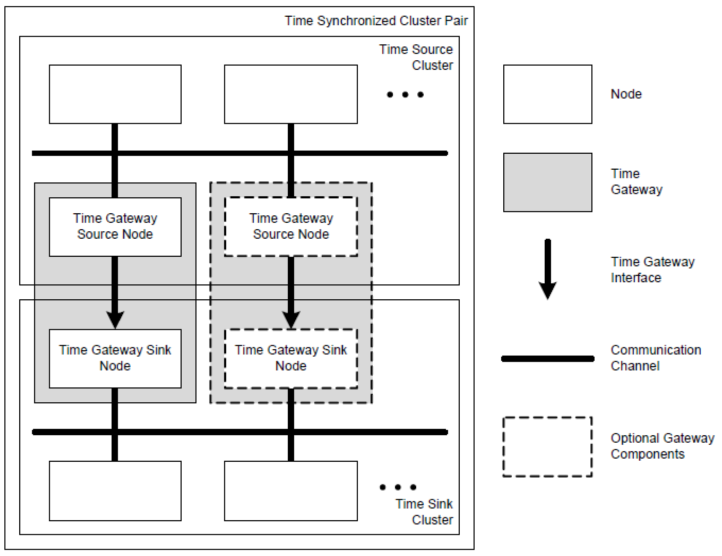

Figure 3: Time synchronized cluster pair.

The advantage of the TT-E synchronization method is the close coupling of the schedule of a TT-E cluster to another FlexRay cluster. In this way, a single FlexRay cluster may be split into synchronized sub-clusters to avoid limits on attached nodes placed upon a single FlexRay cluster by [10] or to enable a separation of nodes into multiple clusters according to communication needs for a more efficient use of the available bandwidth.

1.3.2 Communication controller - bus driver interface The interface between the BD and the CC consists of three digital electrical signals. Two are outputs from the CC (TxD and TxEN) and one is an output from the BD (RxD). The CC uses the TxD (Transmit Data) signal to transfer the actual signal sequence to the BD for transmission onto the communication channel. TxEN (Transmit Data Enable Not) indicates the CC's request to have the bus driver present the data on the TxD line to its corresponding channel. The BD uses the RxD (Receive Data) signal to transfer the actual received signal sequence to the CC. Figure 1.9 shows the connection between the communication controller and the bus driver and the internal connection between the protocol engine and the pins. This protocol specification does not specify a device but the behavior of the FlexRay protocol. In the following the protocol specification only refers to the internal signals TxD, TxEN, and RxD as depicted in Figure 4.

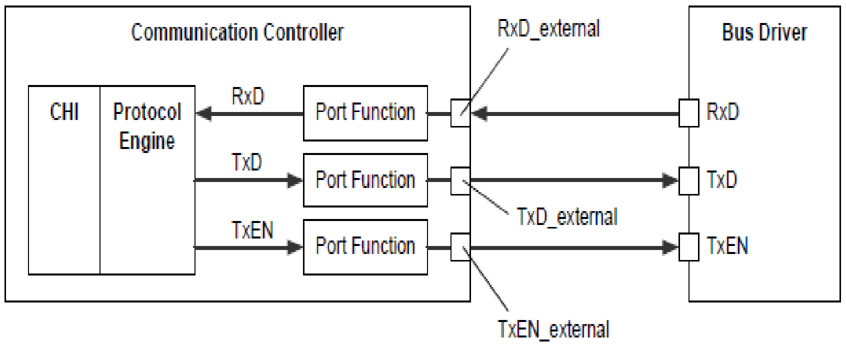

Figure 4: Communication controller - bus driver interface

Between the internal and the external signals there are device specific port functions which are responsible for the electrical behavior of the pins, for example:

• I/O voltage level,
• ESD protection,
• Behavior during power up initialization, reset, or while depowered,
• pin multiplexing (e.g. the connection of the external pins associated with the RxD_external,

TxD_external and TxEN_external signals either to the FlexRay protocol engine (i.e., the RxD, TxD, and TxEN signals, respectively), to some other function inside the CC implementation11, or to nothing at all).

If the pins are connected to something other than the FlexRay protocol engine this specification places no requirements on the behavior of those pins. However, the behavior during power up initialization, reset, while depowered, and the default behavior prior to the configuration of any pin multiplexing, shall ensure that the bus driver does not actively drive the FlexRay bus12, and that the bus driver interprets the TxD signal as low1.

Fujitsu Microelectronics [6] FlexRay also offers many reliability features not available in CAN. Specifically, a redundant communication capability enables fully duplicated network configurations and schedule monitoring by hardware. FlexRay also offers flexible configurations, with support for topologies such as bus, star, and hybrid types Designers can configure distributed systems by combining two or more of these topologies.

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-DESIGN

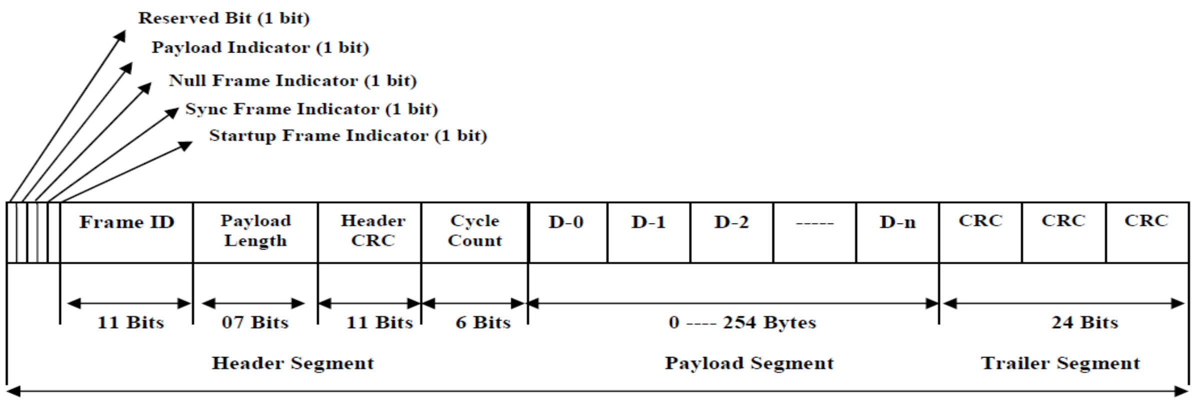

Figure 5. Flex ray Frame Format.

Flex-Ray Frame. Flex-Ray protocol is based on frames, containing data organized in bytes, but transmitted serially. Figure 2 shows the ex-ray frame, consists of 3 seg- mentsVheader, Payload and Trailer. The Header begins with 5 indicators Vthe single bits de_ning basic features of the frame. payload section contains the main data; its length may be variable between 0 and 254 bytes. The Trailer section contains 24 bits of CRC, calculated for the Header and Payload section together. Each cycle is a complex structure, containing static segment, dynamic segment, symbol window and idle time. Static segment the _rst part of cycle contains series of static slots each of these slots is allotted to transmit a single frame. Another part of the cycle is the dynamic segment consists of mini slots. This part of cycle may be used for the frames transmission again but the amount of time allotted a current frame may vary, depending on its length.

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 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

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.

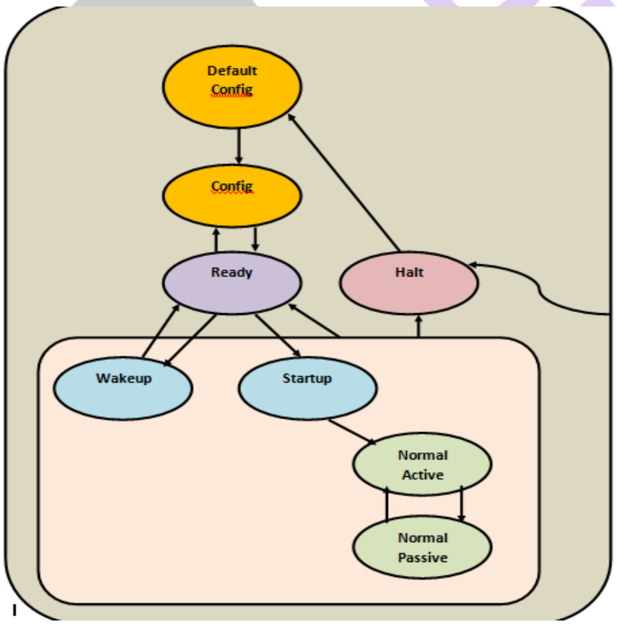

Figure 6 State diagram of FlexRay Protocol Operation Control
In Normal Active state the POC performs Synchronization calculations at the end of each cycle to determine whether the POC should change the modeling of the core mechanisms before the beginning of the next communication cycle. The state changes 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).
FAULT TOLERANT MIDPOINT (FTM) ALGORITHM: The functional principle of the Fault Tolerant Midpoint (FTM) algorithm is as follows. Once the measurements are taken and the divergence values are calculated for both rate and phase differences, then these values are arranged in descending order. Then the most extreme values ‘k’ values from these measured divergences are removed. The algorithm determines the value of a parameter ‘k’, based on the details given in table 2. The remaining one value from both the lists is then considered as the final rate correction value and final phase correction value.

No of sync nodes	Parameter K
1 to 2	0
3 to 7	1
Greater than 7	2

Table 2 FTM ALGORITHM ‘K’ VALUE.

III-RESULTS

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 6 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 4.3 it has all connections & all internal module available means correct coding & design is synthesis properly no any open element found in design.

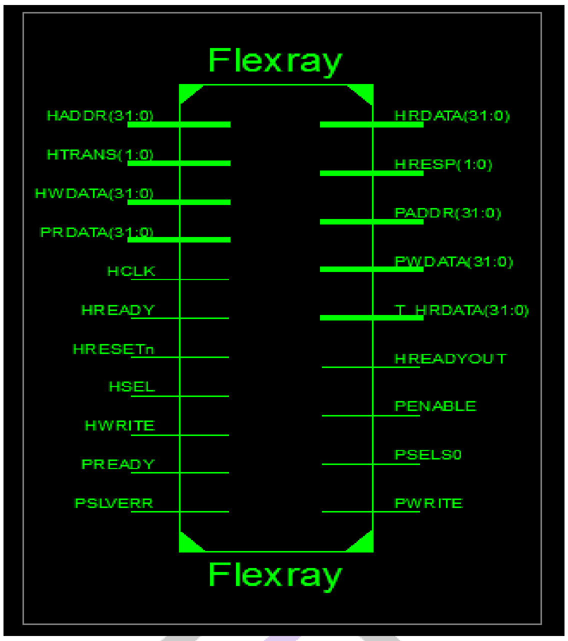

Figure 6 RTL top view

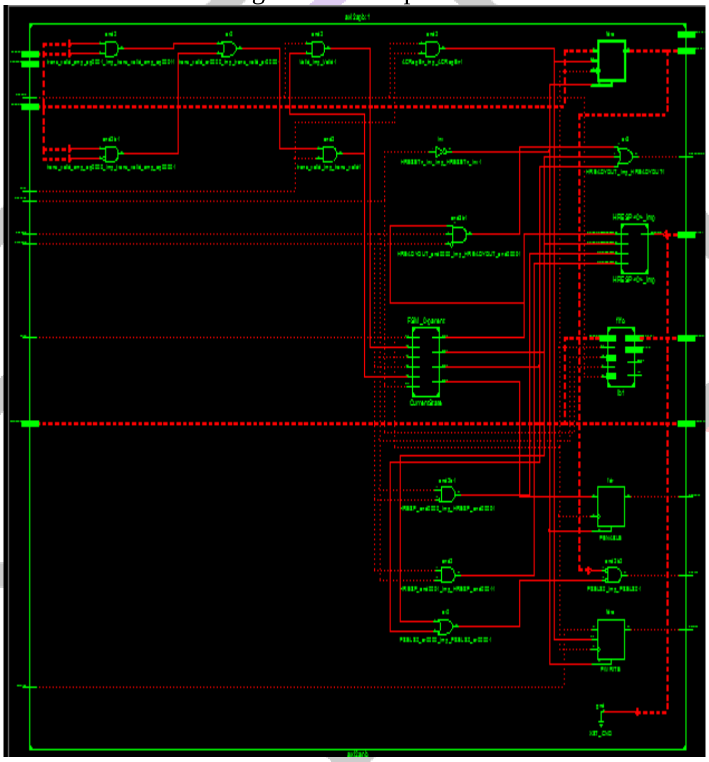
Figure 7 RTL internal view

	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 4 Comparative results

Form table 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

The aim of this paper was to understand the concept and to provide a little contribution towards the discussion on the FlexRay Protocol operation and Clock synchronization process within a cluster. The POC module uses (Finite State Machine) FSM methodology to transfer its control from one state to another. It reacts to the host commands and protocol conditions by triggering coherent changes to core mechanisms in a synchronous manner, and provides the host with the appropriate status regarding these changes. The simulation model for clock synchronization aims at the assessment of FlexRay clock synchronization algorithm (FTM) performance, by means of simulation experiments. The FlexRay clock synchronization algorithm is very well suited to solve the clock synchronization problem in the presence of changing clock drift rates with regard to the achievable precision within a single cluster.

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.
[11] FlexRay Communications System: Protocol Speci_cation Version 2.1 Revision; FlexRay; Consortium, 2006.
[12] FlexRay Communications System Data Link Layer Conformance Test Specification Version 2.1.1
[13] ISE 9.1i Release Notes and Installation Guide, Xilinx Corporation.
[14] TTP/C-C2 communication controller-preliminary data sheet, Rev. 16, May 2002.
[15] Message Handling Concept for a FlexRay Communication Controller-Special Edition FlexRay automotive, 2004.
[16] Multiplexed Networks for Embedded Systems CAN, LIN, FlexRay, Safe-by-Wire, John Wiley & Sons Ltd, 2007.