International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 06, June - 2017

Modeling Integration of CAN and FlexRay Communication Controllers

[1] Pramod B
PG Student, BIT
Bengaluru,

[2]Mrs.Veena H S
Associate Professor, Department of ECE
BIT,Bengaluru

***Abstract*—The Controller area network completely addressed the requirements of the automotive networking until the last decade. However with advancement in the automotive electronics puts requirement of a deterministic and fault tolerant communication systems. Flexray was introduced to address this issue. But Flexray is costly compared to the CAN. Flexray is not seen as the complete replacement for CAN. The current automotives use a mix of communication protocols. There is an occasional need for data exchange between these two protocols for which gateways are required. But gateways are costly. Communication controllers are the core of the data link layer and required for a connection to these networks. In gateway implementations, usually a host that is connected to the two networks does the data conversion in application . Here, a model is proposed at system level for integrating these controllers for achieving host independent interprotocol communication, without the use of gateways and no intervention from the application. An open core IP for CAN CC is used as reference, and for Flexray CC, a simple model, mostly serving as a black box is built taking NXP's MFR4310 as reference.**

Keywords—Controller area network, Flexray, Communication controller, Gateways, Controller architecture, Very Large Scale Integration (VLSI), Communication protocol

I. INTRODUCTION

The introduction of electronics in automobile industry was primarily aimed at increasing the efficiency of engines. Hence the term Engine control unit was introduced. With advancement of electronics for other purposes in automobiles, it was changed to more generic term as Electronic control units. As the functionalities were added the increased complexity could not be handled by one system and hence they were split into a distributed network of systems. However these units still needed to exchange the data between themselves. In the beginning, independent ECUs were sufficient to perform electronic functions. Earlier systems used a point to point communication for data exchange, as they needed to co-ordinate their functionality every signal was allocated a specific channel. However as the number of these systems, wiring required for these connections became also grew and became a major challenge for further additions of electronics. Robert Bosch came up with the solution in 1986 that revolutionized the automotive electronics industry and the Controller Area Network still dominates the vehicular networking domain. CAN is a serial communication protocol employing a single bus. The CAN protocol specification describes the Data Link Layer that includes Medium Access Control, Logical Link Control. CAN reduced wiring and space requirements Though new ECU designs are pushing the limits of CAN communication protocol, and new advanced protocols, such as, Flexray etc are set to make a change, there is no replacement of CAN protocol in the automotive domain for the foreseeable future due to the low cost of implementation, tried and tested operation of CAN protocol in all these years. The usage of CAN has been extended to home and industrial automation, avionics etc. However the complexity of electronics has not stalled and it has become feasible to integrate more and more functions into the system. Examples of these are the X by wire systems. These kinds of applications bring new set of problems and safety risks that cannot be handled by the age old CAN network. CAN network is inherently asynchronous and hence not deterministic in nature. This non-determinism can cause problems in safety critical applications and hence a time triggered access to the communication channel is required which offers more determinism. The FlexRay was developed to address these issues by the FlexRay consortium. However being a new and costly technology, it is facing its own set of challenges to completely be adopted. Modern vehicles have different protocols operating in different regions of the vehicle depending on requirements. When the exchange of data is required between any two protocols, gateways are used. Existing solution for interprotocol data exchange is use of gateways. Gateways use software for processing the data from one network and transmit on other network. This means the gateway operation is handled in the software rather than the hardware. There are already many automotive MCUs incorporating number of both Flexray and CAN network interfaces, though they are handled as separate peripherals. One approach for eliminating the gateways would be to have a node that is connected to both the networks process the information and place it on the other network. But these add to the software complexity and to the processing overheads. This may not always be feasible since an ECU would be dedicated to a function in the network and adding software to handle the data not required by that function will have to account for the effect on that node's requirements. For example the node may have its own deadlines which cannot be interrupted by the data transfer requests. It is also observed that very little amount of data need to transferred from one protocol to another(Ex: Steering angle data present on Flexray is needed in airbag ECU connected to CAN network for recording crash data.), since these are networked within their own group of functions such as power train, body etc.

II. GATEWAY IN LITERATURE

A gateway is a device used to provide communication between networks. This can be between networks using the same protocol, or between networks running on different protocols. The differences in the CAN and Flexray protocol can be found in the fundamental operating principle. CAN is an asynchronous, event driven, protocol with upto 1MBps of data rate [1]. Flexray is synchronous, deterministic, time driven protocol supporting upto 10MBps of data rate [2][3][4]. Hence any gateway system can theoretically achieve a maximum data transfer rate limited by the CAN protocol bandwidth. A CAN to Flexray gateway that is described in the literature uses a framework for gateway design. It is implemented on a microprocessor. The implemented system obtains data from a FlexRay node via the FlexRay bus and translates the data to the CAN protocol. The gateway consists of a standard processor, internal memory and the relevant communication controllers designed at the service level where the networks communicate by directly mapping the services of one protocol to the other. For each decoded message frame it receives, the gateway has to issue the corresponding message frame to the service at the other side for coding and retransmission to the receiving network. Once the message arrives to the message buffer the CPU then takes the information from the message buffer and stores it on its on-board memory [6]. The above implementation uses software to do the data processing. The main advantage observed was that any signal from one network can be mapped to any other signal in other network. This flexibility comes at the cost of software service dedicated to the above task in a dedicated Control unit. It is inferred that there will be no strict requirement of mapping a signal of one network to the one in the other network at the gateway level. Rather the system can be designed such a way that receiving nodes interested in such data can extract it into its application data, thereby moving the configuration requirement away from the gateway to specific nodes that needs this data. Although, today's vehicular networking uses a communication matrix of different protocols for different functionalities, the data exchange between these protocols does not require dedicated transfers but an occasional interruption[7][8][14].

IV. PROPOSED MODEL

The proposed model tries to integrate both Flexray and CAN communication controllers. It also aims to explore the optimization gains, and the cost of integration. It does not focus on the design of these controllers itself, since in most system designs, it is preferred to use already available IPs ,which are proven and verified, and most systems have already individual protocol controllers within them. The objective here is to reduce the interface between the host and controllers, and offload the inter-protocol data transfer from the host, thus eliminating the need for separate gateways. It should be noted here that the proposed model assumes separate stand alone chips for communication controllers. However this model is still applicable in terms of hardware implementation of the data conversion. A simple algorithm is used for transferring data from message of one protocol to the message of another protocol according to some pre-configured rules.

A. Methodology

The architectures of different communication controllers of both the protocol were analyzed. A reference models were chosen for both the types. The behavior of the controllers was modeled at a higher abstraction and with lowest configuration options in order to simplify the model. Any communication with the host has to go through the interface management block. Specific requests for data transfer from the host are automatically routed to corresponding block. The algorithm for inter protocol data conversion is kept simple, assuming there is no requirement of mapping a signal of one network to other signal/Message of other network.

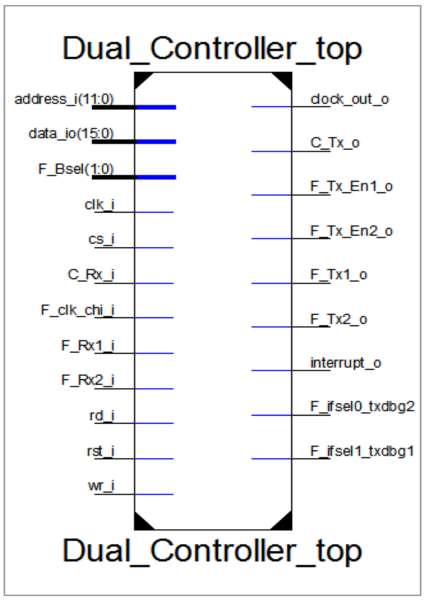

Fig.1. The top module

Only the basic configuration before the communication start is required from the host during initialization phase. A top down approach is followed.

B. Dual controller blocks

Three clock domains are identified as CAN block, Flexray block and third domain belonging to the top module with data conversion tables, few registers and message buffers. There are 5 blocks in the top module. CAN and FlexRay core, Interface management, Clock and reset control and registers.

i) Interface management block**:** The interface management block handles the routing of address, data and the bus signals to respective cores.

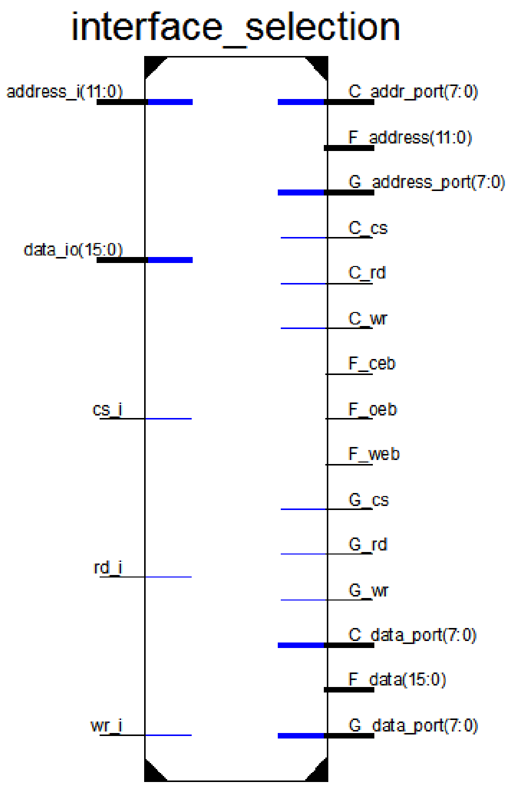

Fig.2. Interface selection block

CAN uses 8 bit address, while Flexray use 12 bit address, although the memory mapped registers are addressed only upto 0x4FE utilizing only 11 bits. The internal 6Kbytes of FIFO and message buffers are accessed using dedicated index, size or field registers. The extra 1 bit at the top module can be used for recognizing Flexray and CAN address space. If the FlexRay address space is to be scaled, then one more address bit can be added to the top module in order to separate CAN and Flexray register accesses. The 11th bit is used for recognizing between the top module registers and CAN registers when the A12 bit is set. Hence when Flexray registers are to be configured, proper addresses are latched to FlexRay module directly. This is identified by 0 values in the 12th address position. When CAN registers are to be accessed, the A12 bit is set while A11 bit is reset, and address is made available at lower 8 bits of the address port. The remaining pins are ignored. While accessing the registers of the top module, A12 and A11 bits are set and lower 8 bits of address is used. When the CAN address space is accessed, the interface management block routes the lower 8 bit of top address pins to address port of CAN module. The bus signals, write enable, chip select and read enable are also routed to the corresponding pins in the CAN module. This happens in a similar way for other modules.

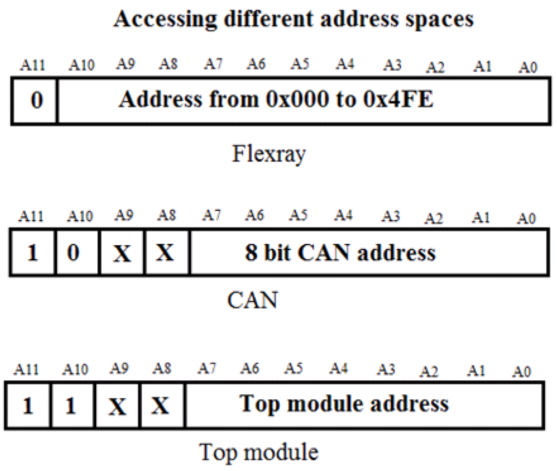
Accessing different address spaces
A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
0
Address from 0x000 to 0x4FE
Flexray
A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
1 0 X X
8 bit CAN address
CAN
A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
1 1 X X
Top module address
Top module

Fig.3. Address spaces of different modules

The address and data port of non selected modules are left floating whereas the bus signals are tied low except for FlexRay module which are active low signals.

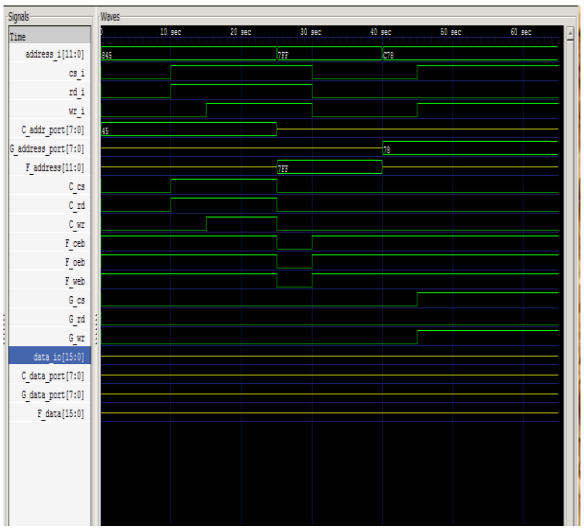

Fig.4. Interface management block test inputs and outputs

ii)Clock and reset control: Has the control for slecting clock for different modules, clockpout pin assignment, and reset generation for top module. The software resets of CAN and flexray module can be accessed by their corresponding address space. This block serves to isolate all the clock related controls of the top module.
iii)Interrupt management: The interrupt line from CAN and Flexray blocks are multiplexed and fed to host. To differentiate between the source, a register is added which has flags inicating the interrupt source as CAN or Flexray. Also the gateway operation events can be enabled for interrupt generation and is indicated by the interrupt control and status registers in this block.
iv)Register bank: It has the registers that are part of the top module. It is served by the clock domain of G_clock.

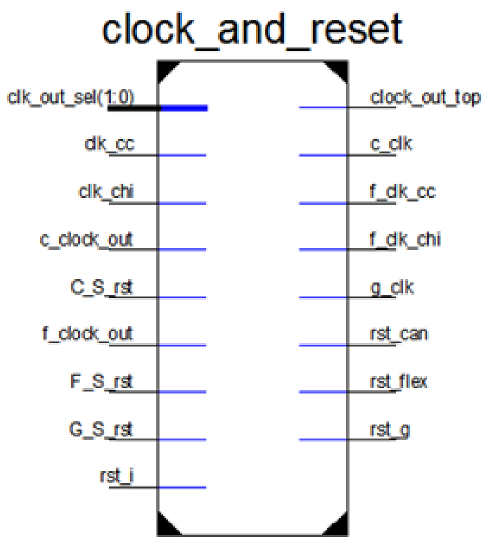

Fig.5. Clock and reset generation block

4.3 Algorithm for data conversion: CAN to Flexray

Eight number of Message receive buffer is implemented for gateway operation. There may be cases where a CAN message has only 2 bytes of data to be transmitted. Or there may be data that is more than 8 bytes and is split into multiple messages by the Application in one of the CAN node.

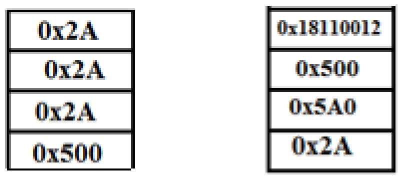

Fig.6. Sample messages in the CAN gateway buffer

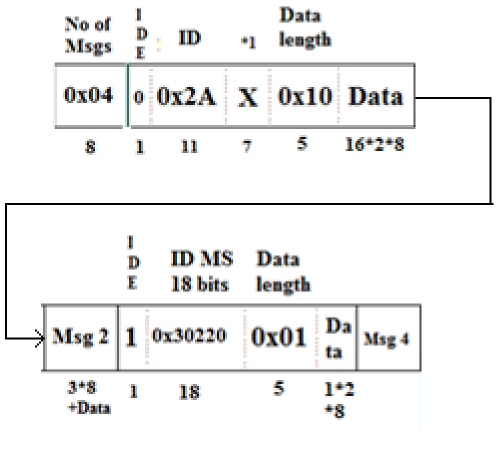

Fig.7. CAN Data packing in FlexRay slot

In order to transmit as many message data as possible in one configured slot, many messages are appended to be accommodated in the configured payload length of the slot. Each ids may also be mapped to a different slot. Also the message may be either transmitted in static or dynamic segment. The frame format for both static and dynamic frame is same. In order to identify the number of messages, Extra information is added in the Payload of the message. The first byte of the message indicates the number of CAN messages of unique ids that are present in the payload. Then each message follows. No particular ordering of message id is followed and order is event based. Next for each message, 3 bytes are used to indicate standard or extended, message id and the data length of the message in the written order. If there were multiple messages of same Id from CAN network, they are all appended and the data length information reflects the total number of bytes in the message. The first bit of the message information indicates whether it is a standard or extended id message. For standard id messages, the full id is sent after the IDE bit. For extended id messages only the most significant 15 bits are sent At the best case of 1Mbps the CAN can generate about 64KBps of actual data, i.e 64KB for every one second. In Flexray at the worst case data rate of 2.5Mbps, if a utilization of 46% is assumed, the data throughput will be 143kBps. Hence the Flexray Network can take away the data faster than the CAN network can produce during normal scenarios. Hence having large number of CAN receive buffers is a waste of resources. The throughput of the gateway operation also depends on the number of slots used and total number of slots configured in the communication cycle. A table mapping the CAN message ids, data to be transmitted, whether to append the data to already arrived message of same id or to write it into message buffer directly, and the message buffer to be which the message has to be transmitted to is implemented. There are 8 table entries, each having a size of 3 bytes. The format of the entry varies according to 29 bit or 11 bit ID CAN message. The first byte always represents the MSB 8 bits of either 11 bit or 29 bit ID (Only total MSB 18 bit considered in entries). The first 3 bits of next byte contains LSB 3 bits of 11 bit ID or bits [21:19] of 29 bit ID. The next 3 bits are ignored for 11 bit identifier and is bits [18:16] of 29 bit ID. The next two bits contain Start byte position/2 of the 11 bit ID to start extraction and bits [15:14] for 29 bit ID. It should be noted that only even position is considered for extraction for both Start and End byte positions. If the DLC of the CAN frame is less than configured bytes to be extracted or is odd, additional bytes are padded. The first two bits of the 3rd byte are the End byte position for the 11 bit ID and bits [13:12] for 29 bit identifier. For 29 bit IDs, always all the data available is extracted and sent. The next bit specifies whether to append data available from multiple messages into a single pack and sent with the ID in the configured slot. The Enable bit follows next. Only if it is set, the entry will be considered for transmission into the Flexray network. The slot information, length etc are not configured in this table. Only the Message buffer to where this data is to be copied is configured in the table. Remaining information such as slot assignment, cycle counter, static or dynamic segment assignment (taken care while assigning the slot or frame id since the first dynamic segment frame id is last static segment frame id+1) are according to the message buffer configuration. Hence these have to be configured in the Flexray module by the host while in the CONFIG state before the start of the communication.

4.4 Algorithm for FlexRay to CAN

A FlexRay message is typically larger than the data that can be fit into a single CAN message. Hence the data is split into multiple messages. The message buffer is configured to receive the data as any normal receive buffers with corresponding filters. The table mapping now contains the buffer number to which a specific CAN message id is mapped. The start and end byte position of the data in the payload is also configured. In case of buffer overruns, an interrupt is generated and the flag is set in the interrupt management block register. Care must be taken as to not exceed the bandwidth limitation of CAN or to flood the CAN network with gateway messages as it may increase the bus load and disrupt the normal operation of CAN network.

V. CONCLUSION

Integration of protocol controllers into microcontroller reduces the cost. Similarly integrating the protocol controllers into a single chip will also reduce the overall area required. The Flexray controller complexity is greater compared to CAN controllers, it may be possible to use some of the blocks already available in Flexray block, such as clock control units, pre-scalars, transmit & receive buffers, reducing devices required compared to individual implementation. Integrating these controller IPs will also result in reduction of overall area requirement if implemented on a SoC, since there is potential of re-using some of the blocks. Integration will also help in simpler and quicker data routing since the data need not travel through the application of the host and can be handled completely in the hardware. A host can use a common interface accessing both the networks. This will reduce the pin count requirement, which are usually at a premium. Since the interface for a FlexRay controller will usually require a faster interface between host and controller to handle data rates of 10MBps, the CAN traffic will not cause excessive overhead and hence can use the same interface. The above model should serve to eliminate the dependence on gateways and instead the operation be transferred to one of the nodes in CAN or FlexRay, with just an additional controller replacement.

REFERENCES

[1] ISO. Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signaling. 11898-1:2003.
[2] FlexRay Consortium. FlexRay Communications System, Electrical Physical Layer Specification, Version 2.1, May 2005.
[3] FlexRay Consortium. FlexRay Communications System, Preliminary Central Bus Guardian Specification, Version 2.0.9, December 2005.
[4] FlexRay Consortium. FlexRay Communications System, Protocol Specification Version 3.0.1, Oct. 2010.
[5] MFR4310 Reference Manual from Freescale semiconductors.
[6] Investigation of a Flexray - CAN Gateway in the Implementation of Vehicle Speed Control ,Brian Somers, Waterford Institute Of Technology.
[7] Intra-Vehicle Networks: A Review Shane Tuohy, Martin Glavin,et al. IEEE Transactions On Intelligent Transportation Systems.
[8] In-vehicle communication networks - a historical perspective and review, Nicolas Navet1, Françoise Simonot-Lion, Proceedings of the IEEE, special issue on Industrial Communications Systems, vol 96.
[9] R. Makowitz, and C. Temple, "FlexRay- A Communication Network for Automotive Control Systems," in proc. IEEE Int. Workshop Factory Commun. Syst.
[10] H. Zinner, J. Noebauer, T. Gallner, J. Seitz, and T. Waas, “Application and realization of gateways between conventional automotive and IP/Ethernet-based networks,” in proc. 48th Design Automation Conference (DAC’11), 2011,
[11] Road vehicles – Unified diagnostic service (UDS), ISO standard 14229.
[12] Gateway Framework for In-Vehicle Networks based on CAN, FlexRay and Ethernet Jin Ho Kim, Suk-Hyun Seo, Nguyen Tien Hai, Bo Mu Cheon, Young
Seo Lee, and Jae Wook Jeon, IEEE Transactions on Vehicular Technology
[13] E-Ray-FlexRay IP-Module User’s Manual from Bosch.
[14] The Potential of CAN FD Technology to Impact Upon FlexRay C. Quigley, A. Williams, R. McLaughlin, Warwick Control Technologies Ltd., CAN in Automation
[15] Gateway System for CAN and FlexRay in Automotive ECU Networks, Zhao Rui, Qin Gui he, Liu, Jia qiao, 2010 International Conference on Information, Networking and Automation (ICINA).
[16] Design of flexray communication controller protocol for an automotive application, IEEE Sponsored 9th International Conference on Intelligent Systems and Control (ISCO)2015, R.Radhiga, J.Pradeep